Low latency group communication over broadcast erasure channels

Mark Andrew Graham

A dissertation submitted to the University of Bristol in accordance with the requirements for award of Doctor of Philosophy in the Faculty of Engineering

School of Computer Science, Electrical and Electronic Engineering, and Engineering Maths

January 2021

Word count: 33180.

This page is left intentionally blank

Abstract

Group communication allows a collection of transceivers to mutually share messages, from either a single or multiple sources to multiple sinks. This form of communication has numerous applications, including vehicular networking, content distribution and multimedia streaming. Reliability and low latency are highly desirable for these applications. This thesis addresses the problem of low latency group communication by developing lightweight distributed algorithms. Most previous work in this area focuses on wired point-to-point links. Motivated by the increased prevalence of wireless networks, this thesis focuses on broadcast channels.

The first problem addressed in this thesis is broadcasting from a single source to large numbers of receivers. The channels to distinct receivers are assumed to be independent discrete memoryless erasure channels. Random Linear Network Coding (RLNC) is shown to achieve non-vanishing throughput, unachievable by ARQ, in exchange for delay scaling with the number of receivers n as $O(\log(n))$. The trade-off between its throughput and delay is quantified.

The next problem considered is all-to-all (allcast) communication in which agents take it in turns to broadcast packets. Each broadcast is received correctly by only a subset of the agents, and is erased at the others. The network of correct receptions at each time-step is modelled by a random graph, and these graphs are correlated over time.

Three algorithms to address this problem are proposed: one in which messages are randomly forwarded, and two RLNC algorithms in which random linear combinations of messages are broadcast. A rigorous mathematical analysis of these algorithms is presented for graphs which are constant over time. The RLNC algorithms are shown to complete in a constant number of time-steps, as opposed to $O(\log(n))$ for the baseline, enabling better scalability to large networks. Finally the analysis is supplemented by simulations, which address the more general setting of graphs evolving as Markov chains.

This page is left intentionally blank

Dedication and Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/L016656], Roke Manor and the EPSRC Centre for Doctoral Training in Communications in the School of Computer Science, Electrical and Electronic Engineering, and Engineering Maths at the University of Bristol. I would also like to thank the School of Mathematics at the University of Bristol for their support.

I would like to thank my supervisors Ayalvadi Ganesh and Robert Piechocki, for their guidance and support throughout my studies at Bristol. The opportunity to work on a project supervised by members of different faculties is rare indeed, and the breadth of knowledge led to rich research problems. I would also like to thank Steve Wales of Roke Manor Research Limited for helpful discussions and industrial insight.

I am extremely grateful to have undertaken The Center for Doctoral Training in Communications programme, which equipped me with the skills to work in such an interdisciplinary area, and afforded me the opportunity. I would like to thank the organisers and staff for their hard work which made this possible, and enriched my doctoral studies.

Finally, I would like to thank my family and friends for their support.

This page is left intentionally blank

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: Mark Andrew Graham DATE: Monday, 16$^{\text{th}}$ August 2021.

This page is left intentionally blank

Contents

List of Figures

This page is left intentionally blank

List of Algorithms

This page is left intentionally blank

List of notation

$\mathbb{N}$	The set of natural numbers $\{1, 2, \ldots\}$.
$\mathbb{Z}$	The set of integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$.
$\mathbb{R}$	The set of real numbers.
$\mathbb{R}^+$	The set of positive real numbers; $\mathbb{R}^+ = \{x \in \mathbb{R} : x > 0\}$.
$\log(x)$	The natural logarithm, the inverse of the function $x \mapsto e^x$.
$\log_a(x)$	The logarithm to base a; the inverse of the function $x \mapsto a^x$.
Bernoulli(p)	The Bernoulli distribution with success probability p.
Bin(n, p)	The binomial distribution, the distribution of a sum of n Bernoulli trials with success probability p.
Φ	The cumulative density function (CDF) of the standard normal distribution.
$D(f\|\|g)$	The relative entropy or Kullback-Leibler (KL) divergence from f to g.
$D(p; q)$	The relative entropy from a Bernoulli(p) to a Bernoulli(q) distribution.
$\lfloor x \rfloor$	The greatest integer less than or equal to x.
$\lceil x \rceil$	The least integer greater than or equal to x.

This page is left intentionally blank

Source code for Monte Carlo simulations

The source code for the Monte Carlo simulations in Chapters 4, 5, 6 and 7 may be downloaded from `https://github.com/mark-a-graham/phd-thesis-code`

This page is left intentionally blank

Chapter 1

Introduction

Point-to-point digital communication has been studied in depth for almost a century. It is well known that error free communication over a noisy channel is possible at any rate below the channel capacity through the use of an error correcting code [3]. The growth of the internet and wireless networking have seen recent interest in network information theory, as many applications are currently held back by naïve implementations of solutions intended for point to point links. In particular, these solutions do not achieve optimal throughput. The lack of coding at intermediate nodes on multi-hop links limits these methods to the end-to-end channel capacity, rather than the minimum cut [4], and the methods fall foul of avoidable bottlenecks when multicasting. These methods do not take advantage of the broadcast nature of wireless systems, treating it as a problem rather than a benefit.

Two decades of research in *Random Linear Network Coding (RLNC)* have been inspired by the problem detailed in [1] (see Chapter 3), which proved by way of counterexample that uncoded methods do not in general achieve optimal network throughput for multicast, and derived the capacity for information flow across a network. RLNC is a method in which messages are considered to be elements of a vector space over a finite (Galois) field, and coded messages are formed by taking random linear combinations of them. As the data flows across a network, new coded messages may be formed by taking random linear combinations of others. Once a receiver has collected a linearly independent set of coded messages, the messages may be recovered by solving the resulting linear system. Variations of this method have been considered for various problems and models, including unicast and multicast over lossless wired networks [5] and lossy wired and wireless networks [6], for which the method is optimal.

The main advantage of RLNC is that one coded message may be useful to multiple receivers with differing requirements, as rather than simply communicating

information about one message, they may aid in the decoding of a group of messages. This allows the method to outperform Automatic Repeat Request (ARQ) in networks which are broadcast in nature, or where several nodes must share a common link, as re-transmissions are only useful to nodes which miss that particular message. RLNC also exhibits the *rateless* property, meaning that the messages may be recovered from any large enough subset of the transmitted messages. This is in contrast with traditional error correcting codes, in which the coding rate must be fixed in advance, and only a fixed number of redundant messages may be transmitted.

The main disadvantage of RLNC is the computational complexity of decoding. One way of reducing the complexity which has been studied extensively in the literature (see Chapter 3) is to form *sparse* random linear combinations, which are a linear combination of only a relatively small number of messages. Such a method is considered in this thesis.

This thesis considers two group communications problems: the problems of multicast and all-to-all (or allcast) communication over broadcast media. These communication problems arise in a number of application areas, and this chapter continues with a brief overview of two such applications. A more detailed overview of the more abstract problems solved in this thesis is given in Chapter 3.

1.1 Motivating application: Connected and Autonomous vehicles

Connected and Autonomous Vehicles (CAVs) have the potential to revolutionise the transport network, with far reaching effects on society. Emerging solutions promise to mitigate climate change by reducing emissions, and reduce journey times and accident rates by improving the safety and efficiency of the road network. The solutions to CAV problems rely heavily on data sharing between vehicles, and this section begins by introducing some challenges in this space. Arguably these solutions are more reliant on connectivity than autonomy, as several of the problems outlined below can be solved at least in part by improved driver information and communication, which could be provided in connected or semi-autonomous vehicles. Whilst widespread adoption of CAVs may take many years, these challenges could begin to be solved by the next generation of road vehicles.

Vehicular platooning on highways aims to reduce the spacing between vehicles, in order to improve road utilisation, and reduce CO_2 emissions (partly due to reduced air resistance by *slipstreaming*). Driving so close to other vehicles would be unsafe

for humans unaided, however Cooperative Adaptive Cruise Control allows vehicles to cooperatively maintain speed and inter-vehicular spacings safely. Such systems must be carefully designed, however, to avoid *stop and go waves* in traffic; a phenomenon in which vehicular speeds oscillate as human-driven vehicles bunch up and spread out on a road, causing unnecessary congestion and pollution. A method of controlling vehicular acceleration is proposed in [7] which guarantees *string stability*, meaning that perturbations in vehicular position are eliminated asymptotically in time. The solution to their equations is difficult to compute, but they show that an accurate approximation may be obtained by using the speed and acceleration data from a fixed number of surrounding vehicles.

Autonomous vehicles can have an impact on the traffic system as a whole, even if they are relatively sparse amongst regular vehicles. For example, it has been shown that a concentration of CAVs as small as 5% [8] is sufficient to mitigate stop-and-go waves. Experimental results linked to a pollution model showed that a single autonomous vehicle placed amongst 20 or 21 others driving in a circle caused a considerable overall reduction in various forms of harmful emissions, and not just for the autonomous vehicle. Hence, these benefits of CAVs may be realised long before the technology fully penetrates the market.

It can be beneficial for vehicles to share their view of the road conditions. For example, continuously sharing measurements relating to the friction between the wheels and the road can allow vehicles to determine the friction of the road ahead. This can be used to make human drivers aware that the road ahead is slippery/icy, and is essential information for autonomous vehicles. Vehicles may also share their road positions, current and target speeds in order to perform cooperative lane changing. Knowledge of the intentions and actions of surrounding vehicles can clearly improve the safety of these manoeuvres, and through cooperation (rather than greedy behaviour) the road utilisation may also be maximised [9] for all. In addition, vehicles may share the locations of obstacles or hazards, and make manoeuvring decisions to avoid them [10].

Whilst automated vehicles alone may utilise their own vast arrays of sensors, data is often required beyond line of sight for their proper and efficient operation (even if compromised operation is possible without), necessitating Vehicle to Vehicle (V2V) communication [9]. The channel conditions in vehicular networks are harsh. The unpredictable movement of vehicles through terrain, and obstacles (including the vehicles themselves [11]), causes considerable packet loss in vehicular networks [11]. In addition, V2V networks suffer a highly *dynamic topology*: the mobility of the vehicles causes intermittent link viability, as well as network partitioning [12]. De-

spite this, CAV applications require low error rates, high data throughput, and are latency intolerant [13].

Many data streams which modern CAV systems must disseminate have not only limited temporal scope, leading to tight latency constraints, but also limited *spatial* scope. For instance, the spatial relevance of safety data may be limited to just 250m [10], whilst trajectory planning may require data sharing over a few kilometers [13]. The limited spatial scope of CAV data makes the use of Vehicle to Infrastructure (V2I) communication impractical, and one barrier to scalability of CAVs is an effective V2V communication system [13].

1.2 Motivating application: Content delivery

In addition to text-based webpages, the internet is increasingly used for the distribution of large files and data streams: from software and updates to movies, video, teleconferencing and internet radio. However *multicasting*, where data is transmitted over a network to multiple receivers, poses significant challenges. To illustrate this, consider for example the problem experienced when Microsoft first began distributing software and updates (such as Internet Explorer) over the internet. Although every end user was downloading the same data, they each had their own connection to a single Microsoft data centre, each carrying the same data independently. The resulting traffic overwhelmed the networks not only for the entire state in which the data centre resided, but also several entire countries with many downloaders; a phenomenon dubbed the *"midnight madness problem"* [14].

The other challenge besides excessive and wasteful download traffic is that the typical error control method ARQ (as used in, for instance, Transmission Control Protocol (TCP) [15]) does not scale well with the number of receivers. Though there are various adaptations of the method, the central principle is that the transmitter will be informed of missing or received packets (or both) by way of feedback (thus necessitating a feedback channel), and will re-transmit any packets it determines have been lost [15]. As the number of receivers grows, clearly the amount of feedback will overwhelm both the transmitter in terms of complexity and the network in terms of traffic; this is the well known *feedback implosion problem*.

A popular modern solution (with other benefits) is known as a *Content Delivery Network (CDN)*. In order to reduce the amount of traffic at the main server, requests for the data they hold are forwarded to the most suitable node on the CDN. These nodes act to cache and forward the most popular requested content, thereby spreading the load of the server and network across a distributed infrastruc-

ture. Since the nodes are typically closer to the end users (possibly within the data centre of their Internet Service Provider (ISP) [16]), and each node downloads the most popular content only once (potentially outside of peak times), much network traffic is saved [17]. However, the use of multiple unicast connections is still not optimal. CDNs are employed by Microsoft for the distribution of updates and web content [18], as well streaming services such as Netflix [16] and BBC iPlayer [19], to name just a few users.

One more issue with ARQ which is not solved by CDNs will be explored in detail in Chapter 4. Clearly as the number of multicast receivers grows, the probability of a receiver needing a re-transmission of a particular message will also increase. It will be shown that this alone will cause the resulting throughput of ARQ to vanish with the number of receivers, even neglecting the method's other issues.

The *digital fountain* principle is introduced in [20]. The idea is analogous to trying to fill a cup of water by holding it under a fountain; it doesn't matter where you hold it, which droplets of water you catch, or the rate at which you catch them. The glass will fill once enough droplets have been caught. The idea is that from k messages, a large number of redundant messages should be produced, of which any k (or close to k) should be sufficient to recover the original messages. In this way, all multicast solutions based on Forward Error Correction (FEC) (including those published prior to [20]) are simply approximations of a digital fountain. Luby Transform (LT) codes [21] were the first method to achieve this principle, and it is clear that network coding does also. By performing network multicasts (by allowing routers to forward messages as necessary) rather than implementing multiple unicasts, and by implementing a digital fountain, clearly agents may simply listen for long enough to obtain sufficiently many coded messages to decode the original message. This method does not require feedback in principle (or at least vastly reduces the level of feedback required, subject to implementation), and solves the problems outlined above.

A promising implementation of fountain coding for multimedia broadcast known as Multimedia Broadcast/Multicast Services (MBMS) was introduced with the 3G standard. Rather than performing multiple unicast connections, the system uses Internet Protocol (IP) Multicast. Note that if multiple recipients reside in the same cell, the same stream of messages would otherwise be broadcast multiple times, regardless of the fact that they can all be received by the same recipients (which simply discard the other streams). Error control is achieved without feedback using *raptor codes*, a form of fountain coding [22].

1.3 Overview

This chapter concludes with an overview of the thesis. Chapter 2 gives an overview of the mathematical preliminaries used in the rest of the thesis. Some simpler lemmata are also proven, whose results are used frequently in the thesis.

Chapter 3 gives a broad literature review, outlining the multicast problem, its history and previous solutions, and an overview of fountain and network coding. An introduction to gossip algorithms, and other methods to achieve allcast communication are also included.

A study of multicasting over broadcast media is presented in Chapter 4. Whilst it is well known that RLNC is throughput-optimal [6], this throughput may only be achieved by coding over arbitrarily large numbers of messages, incurring delay which may be intolerable in practice. The chapter considers the problem of a single transmitter broadcasting a fixed number of messages to a fixed number of receivers, over independent erasure channels (which, overall, is termed a *broadcast erasure channel*). The tradeoff between throughput and delay is quantified for RLNC., and it is shown that whilst ARQ has minimal delay, the throughput it achieves vanishes with the number of broadcast receivers. Further, it is shown that reliable communication is possible at any fixed rate below capacity, for increasing numbers of receivers n, if the application can tolerate delay which is logarithmic in n. This chapter is an extended version of work published in [23].

The allcast problem is considered in the remainder of the thesis. Chapter 5 introduces the problem. A group of agents are synchronised, so that they act in unison at discrete time-steps. Connectivity between the agents at each time-step is modelled by a random digraph; in each time-step, one agent may communicate with another if their corresponding nodes are adjacent in the graph, independent of other nodes. To model the broadcast problem, at each time-step nodes are forced to broadcast one message, and send the same message to every adjacent node. An analysis for and simulations of an uncoded solution for this problem are presented in this chapter. It is shown that this algorithm requires a number of transmissions which is logarithmic in the size of the graph to achieve dissemination of all messages to all agents. This chapter is an extended version of work published in [24].

Chapter 6 introduces a modified version of RLNC to solve the problem posed in Chapter 5. It is assumed that the agents may form a regular partition which is known to all nodes before transmissions commence. Agents first broadcast their own message, and buffer received messages. Then, each subset of the partition is selected in turn, and each agent broadcasts sparse random linear combinations of

messages received from nodes in this partition. Finally, the agents broadcast sparse random linear combinations of all messages buffered in the first round, until all nodes may decode the entire message. An asymptotic bound on the number of time-steps required by the algorithm is derived, showing that the number of time-steps required exceeds a constant with only vanishing probability, and compared with simulation results. This chapter is an extended version of work published in [24].

The algorithm presented in Chapter 6 is designed mostly for ease of analysis. The separation of nodes into a regular partition may be expensive, impractical, and may limit application to more dynamic network topologies. Chapter 7 introduces a simpler method which obviates this requirement. The agents broadcast their own message in the first round, and buffer these messages for use in coding later. In contrast however, the agents broadcast sparse random linear combinations of their entire buffer in all subsequent transmission rounds. An asymptotic bound for the number of time-steps required is presented, which again shows that the number of time-steps required exceeds a constant with only vanishing probability. This improved scalability could allow systems employing allcast to scale to much larger networks. The proof of this result requires an extensive study of the ranks of non-square sparse random matrices, whose rows are not independent. This proof is inspired by the work of [25], and is the main focus of the chapter. This chapter is an adapted version of a draft article [26].

Chapter 8 introduces and formalises the problem of achieving localised allcast amongst agents positioned in a straight line. The modelassumes an infinite line of agents, in which each agent can communicate with those a fixed number of spaces in either direction, but messages are only relevant to those within a certain distance. A baseline solution is discussed and analysed, and a lower bound on achievability is given. A optimal solution using RLNC is left for further research. Finally, a conclusion to the thesis is given in Chapter 9.

The three articles [23, 24, 26] mentioned above are joint work with my supervisors Ayalvadi Ganesh and Robert Piechocki.

1.4 Main research questions and contributions

The main research questions answered by this thesis are the following:

- It is well known that error free communication is possible over a point to point link is possible at any rate below capacity. Is the same true for broadcast channels? How many messages must be buffered for coding at a time? Buffering incurs delay, especially if partial decoding is impossible. Can delay be reduced

in exchange for a reduction in throughput?

- What is the best way to achieve allcast, to minimise the latency experienced, and number of transmissions which must be made to achieve this. Can a solution be implemented in a fully decentralised manner, without fixed infrastructure or relay stations? Can a solution be found which can mitigate slow fading?

The main contributions of this thesis are listed below, for convenience.

- A quantification of the throughput-delay trade-off of RLNC over broadcast erasure channels.
- A rigorous analysis of the number of time-steps required by two RLNC solutions and one uncoded baseline solution to the allcast problem over Erdős-Rényi random graphs.
- A study of rectangular random matrices whose entries are not i.i.d.
- Extensive Monte-Carlo simulations to provide insight for smaller networks, to investigate the convergence of the asymptotic bounds, and gain insight into RLNC allcast performance on more general models.

Chapter 2

Mathematical preliminaries

This short chapter introduces some mathematical preliminaries and results which will be useful throughout the rest of this thesis. For the sake of clarity and readability, this chapter also includes some elementary and simple results which were felt to clutter the chapter in which they are used, which may otherwise be easily verified by the reader but are included here for completeness.

2.1 The relative entropy function and useful properties thereof

The well known *relative entropy* function, also known as the *Kullback Leibler divergence* or *Kullback Leibler distance*, has a variety of applications, including to information theory. A definition (as in [27]) is given as follows.

Definition 2.1. Let $p(x), q(x)$ be probability mass functions over an alphabet $\mathcal{X}$. The *relative entropy* from p and q is given by

$$D(p||q) = \sum_{x \in \mathcal{X}} p(x) \log \left(\frac{p(x)}{q(x)} \right).$$

Although it can be useful to consider the Kullback Leibler "distance" to be the distance between two distributions, it is not symmetric and does not obey the triangle inequality, and hence is not truly a distance in the sense that it is not a metric [27]. One useful inequality known as Gibbs inequality [28] or the Information inequality [27] is as follows:

Lemma 2.2. *Let $p(x), q(x)$ be probability mass functions over an alphabet $\mathcal{X}$. Then*

$$D(p||q) \geq 0$$

with equality if and only if $p \equiv q$.

For convenience, new notation is now introduced for the relative entropy from one Bernoulli random variable to another, which will be the concern of the next lemma, and which will be used in the next section.

Definition 2.3. Let $\mathcal{X} = \{0, 1\}$, and let $\alpha, \beta \in [0, 1]$. Let p, q be the probability mass functions of Bernoulli random variables with parameters α and β respectively. Then

$$D(\alpha; \beta) = D(p||q).$$

It can easily be verified using Definition 2.1 that $D(\alpha; \beta) = \alpha \log \frac{\alpha}{\beta} + (1-\alpha) \log \frac{1-\alpha}{1-\beta}$.

It will now be shown that this function is monotone in each parameter with the other fixed.

Lemma 2.4. *Let $\alpha, \beta \in [0, 1]$. Suppose β is fixed. Then $D(\alpha; \beta)$ is monotone decreasing in α if $\alpha < \beta$, and monotone increasing if $\alpha > \beta$. Similarly, if α is fixed, then $D(\alpha; \beta)$ is monotone increasing in β if $\alpha < \beta$, and monotone decreasing if $\alpha > \beta$. The point $\alpha = \beta$ is the only stationary point of both functions.*

Proof. The partial derivative of D with respect to α is given by

$$\frac{\partial D}{\partial \alpha} = \log \left(\frac{(1-\beta)\alpha}{\beta(1-\alpha)} \right)$$

The final claim of the lemma is obvious in this case, and since the logarithm is greater than zero exactly when its argument is greater than 1, the gradient will be positive when $\frac{\alpha}{1-\alpha} > \frac{\beta}{1-\beta} \iff \alpha > \beta$, and negative otherwise.

Similarly, the partial derivative of D with respect to β is given by

$$\frac{\partial D}{\partial \beta} = \frac{1-\alpha}{1-\beta} - \frac{\alpha}{\beta}$$

and the second claim may be obtained using elementary algebra. □

2.2 Asymptotic notation and convergence

This section lists some useful definitions and notation, beginning with the following, adapted from [29].

Definition 2.5. Let X_n, $n \in \{1, \ldots, n\}$ be a sequence of random variables. We say that X_n *converges in probability* to a limit X, or equivalently write $X_n \xrightarrow{p} X$, exactly when

$$\lim_{n \longrightarrow \infty} \mathbb{P}(|X_n - X| > \epsilon) = 0$$

for all $\epsilon > 0$.

Definition 2.6. Let A_n be a sequence of events. We say that A *occurs with high probability* exactly when $\lim_{n \longrightarrow \infty} \mathbb{P}(A) = 1$.

Asymptotic notation

This thesis makes frequent use of asymptotic notation, such as the well known *"big oh"* and *"little oh"* (O and o) notation. The definition of this notation is to complex to be included in the list of notation in any useful way. Rather than duplicate their rather excellent explanation of this notation here, the reader is referred to [30] for their definitions.

2.3 Geometric random variables

This section introduces a definition of the geometric distribution, for the sake of clarity and completeness (as the distribution is sometimes defined in terms of failure probabilities). The mean of the distribution is derived, before showing that the maximum of n i.i.d geometric random variables converges in probability to a multiple of $\log(n)$.

The geometric distribution models the number of tosses of a biassed coin before the coin lands on heads. A rigorous definition is as follows [29].

Definition 2.7. A random variable is said to be *geometrically distributed* with success probability p, which we equivalently write as $X \sim \text{Geom}(p)$, exactly when its probability mass function is given by

$$\mathbb{P}(X = x) = (1-p)^{x-1}p.$$

The following is well known [29], but an elementary proof is given for completeness.

Lemma 2.8. *If* $X \sim \text{Geom}(p)$*, then*

$$\mathbb{E}(X) = \frac{1}{p}.$$

Proof. We have

$$\begin{aligned}
\mathbb{E}(X) &= \sum_{x=1}^{\infty} x\mathbb{P}(X = x) \\
&= \sum_{x=1}^{\infty} xp(1-p)^{x-1} \\
&= \sum_{x=0}^{\infty} (x+1)p(1-p)^{x} \\
&= \sum_{i=0}^{\infty} \Big(1 - \sum_{x=0}^{i} p(1-p)^{x}\Big).
\end{aligned}$$

Recognising the inner sum as that of a geometric series, we have

$$\begin{aligned}\mathbb{E}(X) &= \sum_{i=0}^{\infty}\Big(1 - p\frac{1-(1-p)^i}{p}\Big)\\ &= \sum_{i=0}^{\infty}(1-p)^i\\ &= \frac{1}{p},\end{aligned}$$

again recognising an infinite sum of a geometric series. □

Finally, we consider the maximum of a number of geometric distributions. If a number of people are tossing biassed coins, this could model the maximum number of tosses any of them made before landing heads. More relevant to this thesis, if n receivers have probability p of receiving a broadcast message, this models the number of times the message must be repeated before all n receive it.

Lemma 2.9. *Let $X_i \overset{i.i.d}{\sim} \mathrm{Geom}(p)$, $i \in \{1, \ldots, n\}$ be i.i.d geometric random variables. Then*

$$\frac{1}{\log(n)}\max_{i=1}^{n} X_i \xrightarrow{p} \frac{-1}{\log(1-p)}.$$

Proof. Let $X = \max_{i=1}^{n} X_i$. For the lower bound, we have

$$\begin{aligned}\mathbb{P}(X < x) &= \prod_{i=1}^{n}\mathbb{P}(X_i < x)\\ &= \prod_{i=1}^{n}\big(1 - \mathbb{P}(X_i < x)\big)\\ &= \prod_{i=1}^{n}\big(1 - (1-p)^x\big)\\ &\leq \exp\Big(-(1-p)^x n\Big),\end{aligned}$$

where the final inequality follows trivially from the Taylor series for e^x. Hence

$$\mathbb{P}\big(\tfrac{X}{\log(n)} < -\tfrac{1-\epsilon\log(n)}{\log(1-p)}\big) \leq e^{-\epsilon n}. \tag{2.1}$$

For the upper bound, we employ the union bound to obtain

$$\mathbb{P}(X > x) \leq n\mathbb{P}(X_i > x) = n(1-p)^x.$$

Hence

$$\mathbb{P}\big(\tfrac{X}{\log(n)} > -\tfrac{1+\epsilon}{\log(1-p)}\big) \leq n^{-\epsilon}. \tag{2.2}$$

The proof follows by taking limits as $n \longrightarrow \infty$ of the bounds 2.1 and 2.2. □

2.4 Bounds for binomial random variables

This section begins with a useful bound on the tail probabilities of binomial random variables, which is useful throughout the thesis. The result is well known [31, pp. 23-24][32], but a full proof is given for completeness.

Lemma 2.10. *Let $p, q \in (0,1)$, and suppose that X is a binomially distributed random variable with parameters (n,p), which we denote by $X \sim \mathrm{Bin}(n,p)$.*
If $q > p$, then

$$\mathbb{P}(X > nq) \leq \exp\big(-nH(q;p)\big),$$

and if $q < p$,

$$\mathbb{P}(X < nq) \leq \exp\big(-nH(q;p)\big).$$

Proof. Let $X_i \sim \mathrm{Bernoulli}(p)$, $1 \leq i \leq n$, and recall that $\mathrm{Bin}(n,p)$ is the distribution of the sum of these random variables. The moment generating function for each Bernoulli random variable is given by $M_{X_i}(t) = \mathbb{E}(e^{tX_i}) = 1 - p + pe^t$. Let $m_t(q) = e^{-qt}M_{X_i}(t)$, and let

$$m(q) = \inf_t e^{-qt} M_{X_i}(t) \tag{2.3}$$

as in [33]. Taking derivatives of m_t gives

$$\begin{aligned} \frac{\partial m_t}{\partial t} &= e^{-qt}pe^t - qe^{-qt}(1-p+pe^t) \\ &= e^{-qt}\big(p(1-q)e^t + q(p-1)\big) \end{aligned} \tag{2.4}$$

It is clear that m_t attains its minimum value at its only stationary point, which occurs where $t = \log\left(\frac{q(1-p)}{p(1-q)}\right)$. Substituting this into Equation 2.3 gives

$$\begin{aligned} m(q) &= \left(\frac{q(1-p)}{p(1-q)}\right)^{-q}\left(1-p+\frac{q(1-p)}{1-q}\right) \\ &= \left(\frac{q}{p}\right)^{-q}\left(\frac{1-p}{1-q}\right)^{1-q} \\ &= \exp(-D(q;p)). \end{aligned}$$

Next, we apply Theorem 1 of [33], widely known as Chernoff's inequality. If $q > p$, we have $P(X > nq) \leq (m(q))^n = \exp(-nD(q;p))$. Similarly, if $q < p$, we have $P(X < nq) \leq (m(q))^n = \exp(-nD(q;p))$. □

Next, a more elementary bound is introduced showing that the cumulative density function of a binomial random variable is monotone in the success probability. A proof is given by a simple coupling argument (see [29, pp. 127-128] for an explanation of coupling).

Lemma 2.11. *Let $p, q \in [0,1]$. Let $X \sim \text{Bin}(n,p)$, let $Y \sim \text{Bin}(n,q)$. Then $\forall i \in \mathbb{N}$, $0 \leq i \leq n$, if $q < p$,*

$$\mathbb{P}(X \leq i) \leq \mathbb{P}(Y \leq i).$$

Moreover, if $q > p$,

$$\mathbb{P}(X > i) \leq \mathbb{P}(Y > i).$$

Proof. We provide a coupling argument to prove the first claim, as the second follows trivially. Suppose $q < p$. Let X_i', Y_i', $1 \leq i \leq n$, be random variables on the same probability space. Suppose $X_i' \overset{\text{i.i.d}}{\sim} \text{Bernoulli}(p)$, $\mathbb{P}(Y_i' = 1 | X_i' = 1) = \frac{q}{p}$, $\mathbb{P}(Y_i' = 0 | X_i' = 0) = 1$, and that $\mathbb{P}(Y_i' | X_1', \ldots, X_n') = \mathbb{P}(Y_i' | X_i')$. Let $X' = \sum_{i=1}^n X_i'$, let $Y' = \sum_{i=1}^n Y_i'$. Clearly $X' \sim \text{Bin}(n,p)$ and $Y' \sim \text{Bin}(n,q)$. For each i, $\{X' \leq i\} \subseteq \{Y' \leq i\}$, hence $\mathbb{P}(X' \leq i) \leq \mathbb{P}(Y' \leq i)$. By Theorem 4.12(3) of [29], $\mathbb{P}(X \leq i) \leq \mathbb{P}(Y \leq i)$. □

2.5 Graph theoretic fundamentals

Graph theory is used extensively in this thesis to model communication networks. This section gives a brief overview of elementary graph theory, in order to define notation which will be used throughout the thesis.

A graph is a discrete mathematical structure consisting of *vertices*, and *edges* which connect them together [34]. A graph $G = (V, E)$ is specified by its set of vertices V, and its set of edges E. In this thesis, each edge in E is taken to be an ordered pairs of its endpoint vertices, i.e $E \subseteq V \times V$. The graph G is said to be *directed* exactly when for each edge $(u, v) \in E$, there exists a corresponding edge $(v, u) \in E$ in the reverse direction. The graph is said to be *undirected* otherwise. Directed graphs are sometimes known as *digraphs*. In connected graphs, an edge (u, v) begins at its *initial vertex* u, and ends at its *terminal vertex* v. In an undirected graph, u and v are called the *endpoints* of the edge [34]. An undirected graph which contains no self-edges (i.e edges from a node to itself) is said to be *simple*. Note that this notation differs with other authors, which define the edge sets of simple graphs to contain *unordered* pairs of vertices [34] (which merely saves the duplication of edges, at the expense of excluding self-edges).

A graph $G = (V, E)$ for which $E = V \times V$ is said to be *complete*, and is said to be *empty* exactly when $E = \emptyset$. The complete graph with vertices enumerated by natural numbers is commonly written as $K_n = (V, V \times V)$, where $V = \{1, \ldots, n\}$. A graph is said to be *bipartite* exactly when its vertex set can be partitioned into two disjoint, non-empty subsets, such that no two edges within a subset are adjacent [34]. Figure 2.1 gives an example of a bipartite graph.

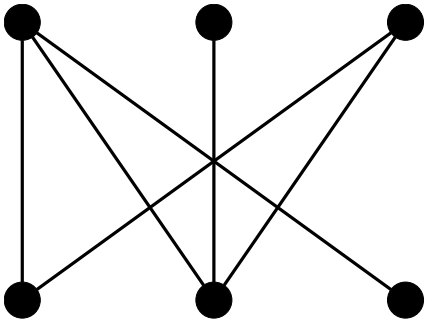


Figure 2.1: An example of a bipartite graph. Notice that the vertices have been arranged into two rows, such that no two to vertices in the same row are adjacent.

Suppose a graph G contains an edge (u, v). If G is directed, then u is said to be *adjacent to* v, and v is said to be *adjacent from* u. If G is undirected, u and v are said to be *adjacent* [34]. For a vertex $u \in V$ of a directed graph, its *in-neighbourhood* $N_u^{\text{in}} = \{(s, t) \in E : t = u\}$ and *out-neighbourhood* $N_u^{\text{out}} = \{(s, t) \in E : s = u\}$ are the sets of all nodes adjacent from and adjacent to u, respectively. The *in-degree* $d_{\text{in}} = |N_u^{\text{in}}|$ and *out-degree* $d_{\text{out}} = |N_u^{\text{in}}|$ are the total number of edges entering and leaving u, respectively. On an undirected graph, the *neighbourhood* N_u of u is the set of all vertices adjacent to u, and its *degree* is the number of such vertices.

A *path* of length n from vertex v_1 to v_2 is a sequence of vertices $v_1, \ldots, v_n$ such that for all $i \in \{1, \ldots, n\}$, $(v_1, v_2) \in E$ [34, adapted]. A directed graph is called *strongly connected* exactly when for each pair a, b of vertices, there exists a path from a to b and a path from b to a. The graph is called *weakly connected* exactly when a path exists in at least one direction between every pair of vertices. An undirected graph is said to be *connected* exactly when a path exists between each pair of vertices [34]. The *diameter* of a graph is the least natural number d such that a path of length d or shorter exists between every pair of vertices in the graph (and in both directions, in the case of directed graphs).

This page is left intentionally blank

Chapter 3

Literature Review

3.1 Broadcasting data: from Automatic Repeat request (ARQ) to Network Coding

This thesis provides analyses of methods for broadcasting data over unreliable media and networks. This section begins by introducing the conventional solution Automatic Repeat Request (ARQ), before illustrating the challenges, history of, and solutions to, the problem area.

3.1.1 Automatic Repeat Request (ARQ)

ARQ is the typical method for correcting errors at the packet level: its popularity extends to 802.11 (Wi-Fi), Ethernet, Bluetooth, Transmission Control Protocol (TCP) and Wi-MAX [15]. Typically, parity data (such as checksum or Cyclic Redundancy Check (CRC) data) is attached to the packet header, so that the receiver may verify the integrity of the packets it receives. Received packets which contain errors are discarded, whilst those deemed intact are acknowledged to the transmitter via a feedback message, commonly known as an "ack". Packets for which the transmitter does not receive a corresponding acknowledgement (ack) are assumed by the transmitter to be lost (for instance due to disconnection or routers shedding packets) or received in error, and re-transmitted. Note that if an ack is lost, an unnecessary retransmission will occur.

If poorly implemented, the feedback process will itself reduce the throughput achieved [15, p.232], particularly if the feedback channel suffers high latency.. In the simplistic "stop and wait" ARQ method, the transmitter is forced to wait until an acknowledgement message has been received before re-transmitting or moving on to the next message. The channel must therefore remain idle after each transmission for

a period of time equal to the round trip time of the feedback channel[15, pp.221-226].

Sliding window protocols mitigate this by weakening this restriction. The transmitter and receiver maintain a window of packets which may be transmitted and received, respectively, so that the transmitter may continue sending messages whilst waiting for previous ones to be acknowledged. This reduces the time the channel remains idle whilst the transmitter awaits feedback messages. An additional efficiency increase is made for bidirectional links by "piggybacking" feedback messages onto outgoing packets [15, pp226-232]. The standard TCP protocol implements a variant of this protocol, and acknowledges the last correctly received message *in order*, discarding any which arrive later (known as *go-back-n ARQ*)[15, pp.565-568]. This reduces the time the channel remains blocked. A commonly implemented TCP option is a form of *selective repeat*, which saves wasting correctly received packets which arrive out of order. The receiver acknowledges ranges of packets to better inform the transmitter of the packets which require re-transmission [15, p.580].

3.1.2 Challenges with broadcast media

One particular application focused on in Chapter 4 is reliable broadcasting over unreliable broadcast media, such as wireless channels. To illustrate this challenge, consider a number of receivers B_i connected to a single transmitter A by a broadcast erasure channel. When A broadcasts a message, each receiver B_i will either receive the message error free, or the message will be irrecoverably lost at random. Suppose the transmitter employs the ARQ method, and that a particular receiver reports messages which are missing to the transmitter, so that they may be re-transmitted. Clearly receivers which receive each message earlier than others are forced to wait several time-steps whilst A broadcasts re-transmissions which aren't useful, wasting precious time-steps [28, pp.593-594]. It is easy to consider circumstances in which coding may alleviate this. For instance, consider the example illustrated by Figure 3.1. Two messages are broadcast to two nodes, one node receives one message, and the other receives the other message. Clearly it would take at least two re-transmissions to complete the multicast in this case, in the absence of coding. However, transmitting the sum of the messages achieves this using one successful transmission.

In this thesis, messages will be considered to be elements of the vector space $\mathbb{F}_q^l$; a vector of l *symbols* from the *alphabet* $\mathbb{F}_q$. For example, we may consider a message of eighty bits to be an element of $\mathbb{F}_2^{80}$. Or we may consider it to be a packet of ten bytes, and an element of $\mathbb{F}_8^{10}$. Using this algebraic structure, we may now take linear combinations of messages over $\mathbb{F}_q$, which we may consider to be *coded*

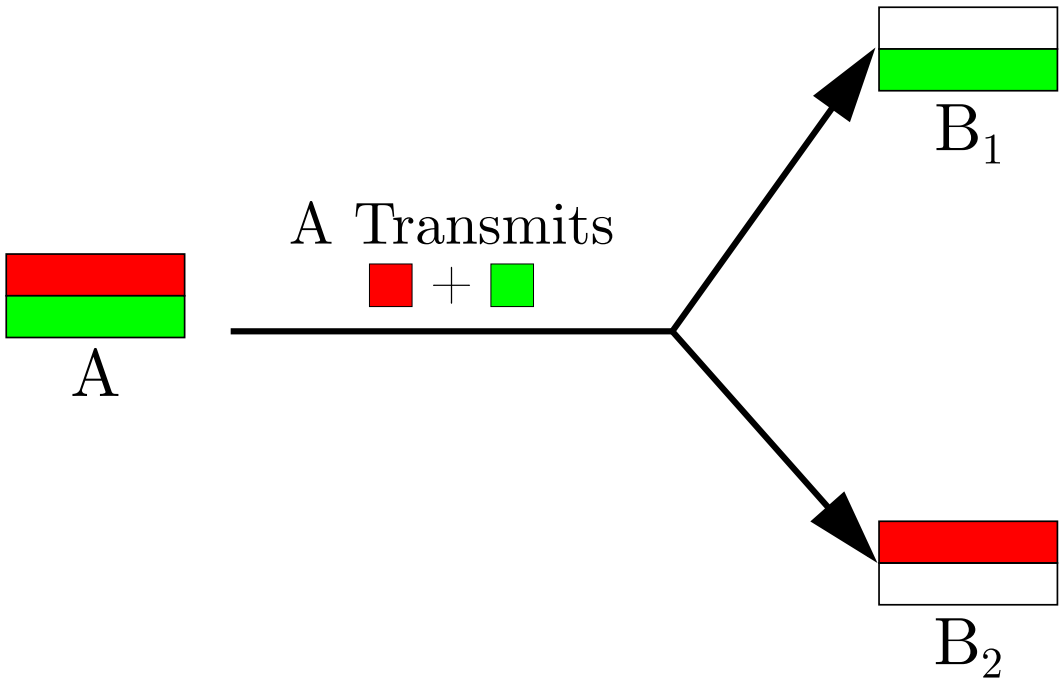


Figure 3.1: Graphic demonstrating the advantage of network coding; that a linear combination of messages may satisfy the diverse requirements of multiple receivers simultaneously. Notice how two transmitters may each receive a distinct message from a single successful transmission.

messages. Whilst these operations have no meaning in terms of the data we wish to communicate itself, this is inconsequential so long as the intended recipients may in some way *decode* the original messages from the coded ones. Returning now to the example: Suppose we consider the messages to be elements of a vector space. If the transmitter were instead to broadcast the sum of the two messages after the second round, then each node may decode the message they are missing after one successful re-transmission. This is in contrast to the previous case, in which one re-transmission may be received by agents which don't need it, but not by those which do. This clearly has the potential to save numerous re-transmissions.

Note that as we cannot predict when and where erasures will occur (or at least not cheaply and with certainty), the method outlined above is clearly not practical. This motivates the randomised design of *fountain coding*, which is as follows. Before coding commences, the transmitter will buffer k symbols from the stream it wishes to communicate. Coded symbols for transmission are formed by taking random linear combinations of the k buffered messages, i.e if we label the buffered and coded messages M_i and C_i, then $C_i = \sum_{j=1}^{k} a_{i,j} M_i$, where $a_{i,j}$ are chosen i.i.d at random from some distribution over $\mathbb{F}_q$.

As each receiver receives linear combinations of the message packets, over time

it will obtain a linear system, which will yield the messages in the transmitter's buffer when solved (which can be achieved, for example, by Gaussian elimination). The receiver must however be in possession of the coefficients used for encoding. The most common method for sharing these coefficients is for them to be generated by the transmitter and included in each transmission as a packet header, whose overhead is generally neglected in analysis due to its size relative to the payload. Alternatively, the transmitter and receiver may generate the same coefficients using identical Pseudo-Random Number Generators (PRNGs), share their seeds and keep them synchronised over time [21]. The term fountain coding is broad, encompassing many different methods which have similar properties and seek to solve the same problems, and arguably the main difference between these are the way in which the $a_{i,j}$ are chosen.

The main advantage of fountain coding is that, so long as the distribution of the $a_{i,j}$ is well chosen, the system at each receiver will be full rank after approximately k messages have been received. Hence, the messages may be decoded from any subset of the coded messages which are broadcast, in any order, so long as enough are received. This is known as the *rateless* property of fountain codes, which are sometimes simply referred to as *rateless codes*. This is in stark contrast to conventional error correcting codes, in which the amount of redundant information must be fixed before transmissions commence. In the case of multicast, this allows each coded message to be useful to a large number of receivers, regardless of which messages they have already received. Other advantages include the ability for a transmitter to broadcast coded messages indefinitely, and for receivers to connect and disconnect at any time, without new receivers having to wait for a scheduled broadcast and listen for its entire duration.

3.1.3 Towards a digital fountain

The feedback required by ARQ is entirely wasteful, as the capacity of a discrete memoryless channel is not increased by the presence of a feedback channel[35]. Moreover, this prevents its application to channels without feedback. ARQ also suffers from the well known *feedback implosion problem*. Clearly with each transmission there must be a corresponding acknowledgement message, meaning that the feedback traffic must scale at least linearly with the number of receivers. This overwhelms not only the feedback channels, but also incurs a considerable computational overhead at the transmitter [36].

Before introducing methods to mitigate these issues, we begin with an overview of Reed-Solomon codes. They are *Maximum Distance Separable (MDS) codes*, as

their minimum distance meets the Singleton bound $d_{\min} \leq n-k+1$ with equality [37, p.79]. It is well known that a code may correct $d_{\min} - 1$ erasures [37, p.79], provided the location of these erasures is known. In practice, this means that all k symbols may be recovered from any k coded symbols [37]. In comparison to Random Linear Network Coding (RLNC), the equivalent $k \times k$ decoding matrix for Reed-Solomon codes will always be invertible, whereas for generally for RLNC it may not be. The expected number of received messages required by RLNC (when coding over $\mathbb{F}_q$) to obtain a linearly independent k-subset is shown by [38] to be upper-bounded by $k\frac{q}{q-1}$. Clearly RLNC approaches the singleton bound for increasing field sizes, and its overhead decays with its blocklength. But for Reed-Solomon codes, the ability to decode from k messages is guaranteed. One advantage of Reed-Solomon codes is that they may be decoded using the Berlekamp-Massey algorithm, with complexity $O((n-k)^2)$ [39]. Note that for an erasure channel, the expected number of erasures will be a fixed fraction of n, and so the complexity is quadratic in n in practice. For a thorough treatment of Reed-Solomon codes, see [37].

Error control using Forward Error Correction (FEC) and ARQ need not take place in complete isolation. In fact, a hybrid of FEC and ARQ known as Hybrid Automatic Repeat reQuest (HARQ) can improve throughput. Precoding messages before using ARQ will reduce the number of re-transmissions required, but some will still be required. Instead of re-transmitting messages, HARQ systems successively poll the receivers after sending the entire message to determine the maximum number of packets lost by the worst receiver. An error correcting code is applied to the message, and a sufficient number of parity messages are broadcast, until all receivers can decode every message [40]. The system explained in [40] employs a *systematic* Reed-Solomon code across a buffer of k packets, which produces an additional $n-k$ packets such that decoding is possible from any k out of the total n. This way, retransmissions are useful for any receivers which have missing packets, rather than just those missing one packet in particular. This process repeats until all parity packets have been sent. The receivers are polled for any outstanding packets, and these are transmitted alone using the same method. This method was shown to outperform a separate FEC-ARQ system in [40], and that both outperform ARQ alone.

A variety of methods aim not only to eliminate feedback, but also reduce the number of redundant re-transmissions received by multicast receivers which have already received the original message (or a re-transmission) correctly. This is achieved by applying some form of FEC to the data (possibly ahead of transmission time), so that one re-transmission my be useful to multiple nodes. The *Microsoft FCAST*

protocol, initially introduced in [14] and then improved in [41], provides reliable network multicast of a fixed number of packets to a number of asynchronous users. The authors exploit Internet Protocol (IP) multicast, which allows receivers to subscribe to a multicast address, and receive broadcast messages for as long as they wish. Messages are replicated and forwarded by routers, and messages are not retransmitted for those that miss or fail to receive them. This saves having multiple copies of the same message passing down the same edge of a network as would be the case for multiple unicast connections.

This great saving in network traffic comes at the cost of foregoing a feedback channel. The authors overcome this by partitioning the file for broadcast into k packets, and then applying a systematic Reed-Solomon code over them before communications commence, to obtain a total of n packets. A receiver which receives any k unique packets may then decode the original file. The authors of [14] then employ the *data carousel approach* to broadcast the data and redundant packets. The transmitter broadcasts each packet in turn, before returning to the first packet and repeating. In the absence of coding, clearly this will allow reliable reception of the message to any receiver eventually. Since each message (coded or otherwise) will be repeated on every cycle of the carousel, a data carousel is effectively a simple repetition code. However, Reed-Solomon coding allows a receiver which experiences fewer than $n - k$ erasures to receive the file in one cycle of the carousel, greatly reducing the number of duplicate messages received, as well as the overall latency.

Nodes which experience more than $n - k$ erasures will be able to wait for subsequent cycles of the carousel, in order to receive a subset of the packets received. This solution is far from ideal; only a fixed amount of redundancy may be introduced, which must be decided upfront, and in general those receivers which have worse channels than expected will receive little benefit. As pointed out in [20], the amount of redundancy which may be introduced using a Reed-Solomon code is limited due to the complexity of decoding it. The authors introduce the concept of a *"digital fountain"*, to which all previous methods (including their proposition) are an approximation. Receiving data from a digital fountain should by like collecting water from a water fountain using a cup; any receiver which receives k packets, in any order, from any source, should be able to decode the original k packets (from their "full cup"). The more redundancy introduced through coding, the longer it takes for the carousel to repeat, and the better an approximation the protocol becomes to a true digital fountain.

The authors of [20] stopped short of realising this principle entirely, but introduced a class of codes which are a much better approximation of a digital fountain,

known as Tornado codes. Encoding is performed according to a random graph with three "layers"; the subgraphs between the first and second and the second and third layers are bipartite. The first layer represents the source symbols, and the others parity symbols. Each parity symbol is the sum over $\mathbb{F}_2$ of the symbols it is adjacent to in the previous layer. Encoding has linear complexity, and decoding is possible in linear time using the belief propagation algorithm detailed in Section 3.2.1. This reduction in complexity makes the generation of large numbers of redundant symbols, for large blocklengths, much more practical. However, this comes at the expense of decoding inefficiency. Whereas a (k, n) Reed-Solomon code may be decoded using any k coded symbols, a Tornado code requires $(1 + \epsilon)k$ symbols before decoding is possible, for some constant $\epsilon \in \mathbb{R}^+$ [20]. The first codes to realise the digital fountain principle are Luby Transform (LT) codes [21], introduced in Section 3.2.1.

3.1.4 Achieving optimal network multicast throughput

The capacity of a *multi-terminal network* is upper-bounded in [27]. The authors show that the flow across a network of M nodes connected arbitrarily by memoryless channels cannot exceed any cut on the network. Similarly to the well known *maximum flow minimum cut* theorem for flows on graphs [42], this shows that the capacity cannot exceed the minimum cut.

The original motivation for RLNC was to improve throughput on static, lossless, wired networks, such as the internet. It has been shown that even for networks with lossless links, the minimum-cut capacity of the network cannot in general be achieved for multicast by uncoded methods, such as storing and forwarding messages at intermediate nodes. Consider the counterexample shown in Figure 3.2, taken from [1]. Nodes in the figure are connected by one-way, error free channels of unit capacity. Node s has a sequence of bits $b_1, b_2, \ldots \in \mathbb{F}_2$ to communicate to both nodes t_1 and t_2 over the network, with the assistance of all other intermediate nodes. The graphic shows how two bits of data may be communicated by using a single transmission per edge. Since each node t_i receives b_i and $b_1 + b_2$, each may recover the bit they did not receive directly by taking the sum of the two received bits. By inspection of the graph, without the use of coding, it is clearly impossible to do the same without coding; indeed bits b_1 and b_2 would need to be transmitted sequentially from u_1 to u_2, at a cost of an extra transmission. Notice again how one transmission is serving two nodes in different ways at the same time.

If Figure 3.2 were to represent a fluid network, we could use the well known max-flow min-cut theorem [42] to show that a maximum of two units of fluid may continuously move from s to t_1 and t_2 *in total*; that is, the sum of the achievable

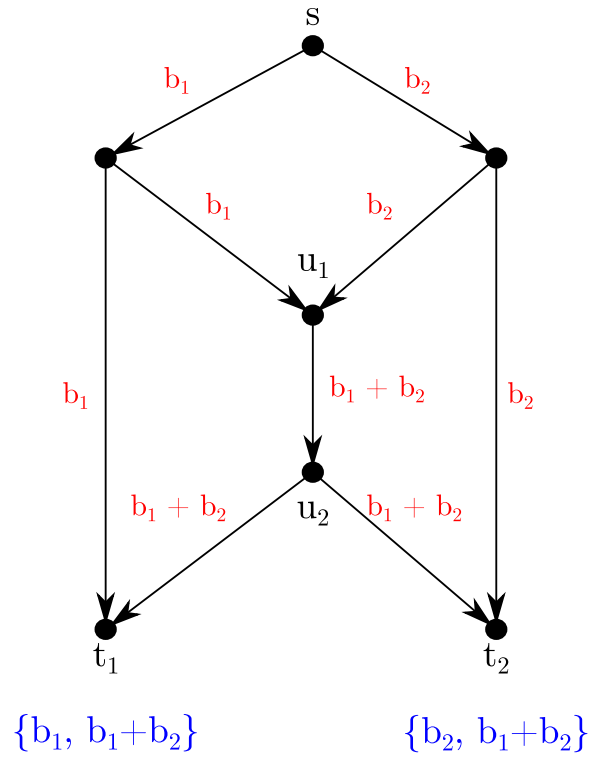


Figure 3.2: Graphic from [1] detailing a network which requires coding to achieve capacity.

flow rates achieved at t_1 and t_2 would be upper bounded by 2. However, the figure represents a data network, whose edges carry bits of information. Clearly the information flow in the figure is in excess of this, and in fact each sink node achieves its maximum flow simultaneously. This flow would not be possible without the network coding system the figure details.

The authors of [1] consider communication over directed graphs containing a single source s and L sink nodes t_l, in which edges represent error free links with fixed (possibly distinct) rates. They give a broad definition of a block code for such networks, which they name α codes. An $(n, (\eta_{ij}, (i,j) \in E), h)$ α-code is defined as follows. The parameters can be considered to be the block-length, a vector containing the amount of information transmitted over each edge (in terms of the total number of possible symbol combinations), and what may be termed the coding rate. The source s chooses a single symbol from the alphabet $\Omega = \{1, \ldots, \lceil 2^{nh} \rceil\}$ with a uniform distribution. The code is constructed from a number of components:

1. An integer K: the total number of *transactions* (defined later).

2. Functions $u : \{1, \ldots, K\} \longrightarrow V$, $v : \{1, \ldots, K\} \longrightarrow V$, such that $(u(k), v(k)) \in E$, to define which edge is involved in each transaction.

3. An alphabet of symbols to transmit in transaction k. This is defined as $A_k = \{1, \ldots, |A_k|\}$, for each $1 \leq k \leq K$, given that the set $|A_k|$ has cardinality greater than 1, and subject to the condition that $\prod_{k \in T_{ij}} |A_k| = \eta_{ij}$, where $T_{ij} = \{1 \leq k \leq K : (u(k), v(k)) = (i, j)\}$. This condition is a consequence of the second parameter.

4. A set of encoding functions $f_k : S_k \longrightarrow A_k$. If $u(k) = s$ (i.e the source is transmitting in transaction k), then $S_k = \omega$. Otherwise, $S_k = \prod_{k' \in Q_k}$, where $Q_k = \{1 \leq k' < k : v(k') = u(k)\}$, the Cartesian product of the alphabets of the previously received symbols at that node.

5. A set of decoding functions $g_t : \prod_{k' \in W_l} A'_k \longrightarrow \Omega$, for $1 \leq l \leq L$, where $W_l = \{1 \leq k \leq K : v(k) = t_l\}$, the set of all transactions in which data is transmitted to t_l.

Communications proceed via a sequence of K transactions in chronological order. In each transaction k, node $u(k)$ encodes according to f_k (in doing so selecting an index from A_k), which it then transmits to $v(k)$. The definition of f_k permits the node $u(k)$ to code across messages it has received by this point.

Let $G = (E, V)$ be a directed graph, let $\mathbf{R} = [R_{ij}, (i, j) \in E]$. A tuple $(\mathbf{R}, h, E)$ is said to be *admissible* exactly when for each $\epsilon > 0$, for sufficiently large n there exists an $(n, (\eta_{ij}, (i, j) \in E), h - \epsilon)$ α-code such that $n^{-1} log_2(\eta_{ij}) < R_{ij} + \epsilon$, for all $(i, j) \in E$. Define $\mathcal{R}_{h,G} = \{\mathbf{R} : (\mathbf{R}, h, G)$ is α-admissible$\}$.

The main theorem of [1] characterises this achivable region. Let $G(E, V)$ be a directed graph, with capacity R_{ij} on each edge (i, j). Let $\mathcal{R}^*_{h,G}$ be the set of all $\mathbf{R}$ such that for each $1 \leq l \leq L$, the maximum fluid flow (i.e minimum cut [42]) from s to t_l is greater than or equal to h.

Theorem 3.1.

$$\mathcal{R}_{h,G} = \mathcal{R}^*_{h,G}$$

This shows that any rate below the minimum cut between the source and each sink node in isolation is achievable, and that there exists no α-code which achieves rates in excess of this. Whilst the authors of [1] acknowledge that more general definitions of block codes could be formulated, they extend their results to prove that randomised coding cannot achieve greater rates. The authors also comment that it is unlikely for there to be any generalisation of their definition of a block code which would permit greater throughput. We may therefore consider this to be the multicast capacity of these networks.

An RLNC solution to this problem is introduced in [5]. There are (possibly many) unit rate sources of information at each node, each of which must be disseminated to a subset of the nodes in the graph (which is assumed to be disjoint with the set of "sources"). There are i sources of information in total. In contrast to Section 3.1.2, buffering is not allowed at intermediate nodes; each node instead maps symbols from incoming edges and information sources at the node, to linear combinations as output symbols to transmit across each outgoing edge (and these mappings need not be identical). Sink nodes map the symbols received on their edges to i *output processes*, as linear combinations of them. The coefficients of these linear combinations are chosen at random from a finite field, and since the network topology is static, they are fixed for all time (although they may be re-generated a few times before fixing them, in order find a suitable solution). We may notice that the i output processes at each node form a system of linear equations. So long as this system is full rank, we may find its inverse, and suitably change the coefficients at the receiving nodes in order to obtain decoded input symbols as output processes. So long as the system is full rank, clearly communication is achieved at the maximum rate derived in [1]. The authors show that this is indeed the case with high probability for increasing field sizes.

The application of RLNC to networks with lossy links was analysed in [6]. The authors model *wireline* networks as directed graphs, and *wireless* networks as hypergraphs: graphs in which each edge may have multiple "endpoints". The authors assume that traffic arrives on each link according to a counting process of constant rate, and that each message that arrives at a node is erased i.i.d with constant probability. The authors define *unicast* connections to be from a source s to a sink t, and *multicast* connections to be from a source s to a set of nodes T. In the method the authors present, the source node buffers k messages that it wishes to transmit, and all other nodes buffer every message they receive. At each transmission opportunity on an edge, the node transmits a random linear combination of messages in their buffer, with coefficients taken uniformly at random from the finite field.

The authors assume that each node may communicate the coefficients of each linear combination by including them as a packet header, with no affect on the rate at which it may communicate the rest of its message, and that the amount of side information will be allowed to grow. This is a relatively standard assumption, and the authors likely avoided alternatives which would obviate the need for this so that their method could remain decentralised. For instance, the coefficients could be generated by a pair of synchronised PRNGs, as mentioned previously. However, it would be necessary for all agents to mutually exchange their seeds in advance of

communications, leading to a costly setup process.

The capacity between two nodes on the network is the minimum cut between them. That is, the maximum flow which may be achieved towards the sink across any separating line in the network. The authors show that for both unicast and multicast communication on wireline and wireless networks, each sink node t may achieve a communication rate arbitrarily close to the minimum cut between s and t using RLNC for large enough k, and hence that the method is capacity achieving.

3.2 Sparse and complexity reducing methods

One challenge for fountain coding and RLNC is the computational complexity of encoding and (mainly) decoding; decoding a k packet message with coding coefficients chosen uniformly from a finite field requires inverting a dense $k \times k$ matrix and multiplying it by a k-vector of coded symbols. The matrix inversion using Gaussian elimination requires $O(k^3)$ operations, but although the order of complexity is lesser, the cost of $O(k^2)$ operations to multiply the inverse dominates in practical situations due to the size of the data packets [43]. Clearly the encoding complexity is also $O(k^2)$.

Several methods have been devised to reduce the complexity of encoding and/or decoding, by choosing the majority of coefficients to be zero (sparse methods), or by choosing coefficients in such a way as to allow for particular, low complexity decoders to operate efficiently and reliably.

3.2.1 Luby Transform (LT) codes

LT codes are a class of low complexity fountain codes first introduced in [21], whose encoder is designed to permit the use of the belief propagation decoding algorithm. Message symbols are assumed to be members of a vector space over $\mathbb{F}_2$, and encoding symbols are generated using random bipartite graphs between input and output symbols, as in Figure 3.3 (a). To generate a new symbol, a new node is added to the graph, and its degree d is selected at random from the *degree distribution*. Then d message packet nodes are selected uniformly at random as neighbours, and edges between them and the new coded symbol are added to the graph. The value of the coded symbol is the sum of the message symbols adjacent to it.

The belief propagation algorithm may decode the message by successively manipulating the graph, as illustrated in Figure 3.3. Clearly if a coded symbol has degree one, then its value is equal to the corresponding message symbol M_i, and may trivially be decoded. Such symbols are illustrated in Figure 3.3 (a), (c) and

(e), with emboldened edges between the message and coded symbols. We may then replace each coded symbol adjacent to M_i by its sum with M_i, eliminating M_i from its associated linear combination, and then remove M_i from the graph. This is illustrated in Figure 3.3 (b), (d) and (f), where the deleted edges are dashed. The deletion of edges may yield additional coded symbols of degree one, and in this case the algorithm may be repeated to decode other symbols.

The belief propagation algorithm will successfully decode the all message symbols so long as there exists at least one coded symbol of degree one in the graph at each stage of the algorithm. This is ensured by choosing the degree distribution such that at each stage of the algorithm, in which a degree-one symbol is always removed, at least one more is likely to appear. The distribution which achieves this is known as the *robust soliton distribution*, which yields symbols of average degree $O\big(\log(\frac{k}{\delta})\big)$, and decoding success probability $1-\delta$ [21].

Both the encoding and decoding require $O\big(k\log(\frac{k}{\delta})\big)$ symbol operations, which is a considerable improvement over Gaussian elimination for decoding, and even encoding by choosing coefficients uniformly. This method pays a small efficiency penalty in exchange for low complexity; the original k symbols are recoverable from any $k+O\big(\sqrt{k}\log^2(\frac{k}{\delta})\big)$ coded symbols with probability $1-\delta$ [21]. This overhead clearly becomes small in comparison to k for large k, and in practical implementations this corresponds to around 5% [28].

3.2.2 Raptor codes

An extension of LT codes, known as *Raptor codes*, are presented in [44]. Whilst the encoding and decoding complexity of LT codes is considered low at $O(k\log(k))$, raptor codes are able to achieve linear encoding and decoding complexity. This comes at the expense however of a decoding overhead; a raptor code requires $(1+\epsilon)$k messages to decode a block of size k, for some constant $\epsilon > 0$. The author shows that in exchange for this overhead, an LT code with well chosen degree distribution may encode the data with linear complexity. Decoding is also possible with linear complexity using the belief propagation algorithm, however a fixed fraction of the messages will be un-recoverable at random. This is overcome by concatenating the LT code with a *precode*. The author shows that the concatenation of a well chosen Low Density Parity Check (LDPC) code and LT code have linear encoding and decoding complexity when using the belief propagation algorithm to decode both codes. The overhead of the code can be made arbitrarily small, in the sense that ϵ can be made arbitrarily close to 0.

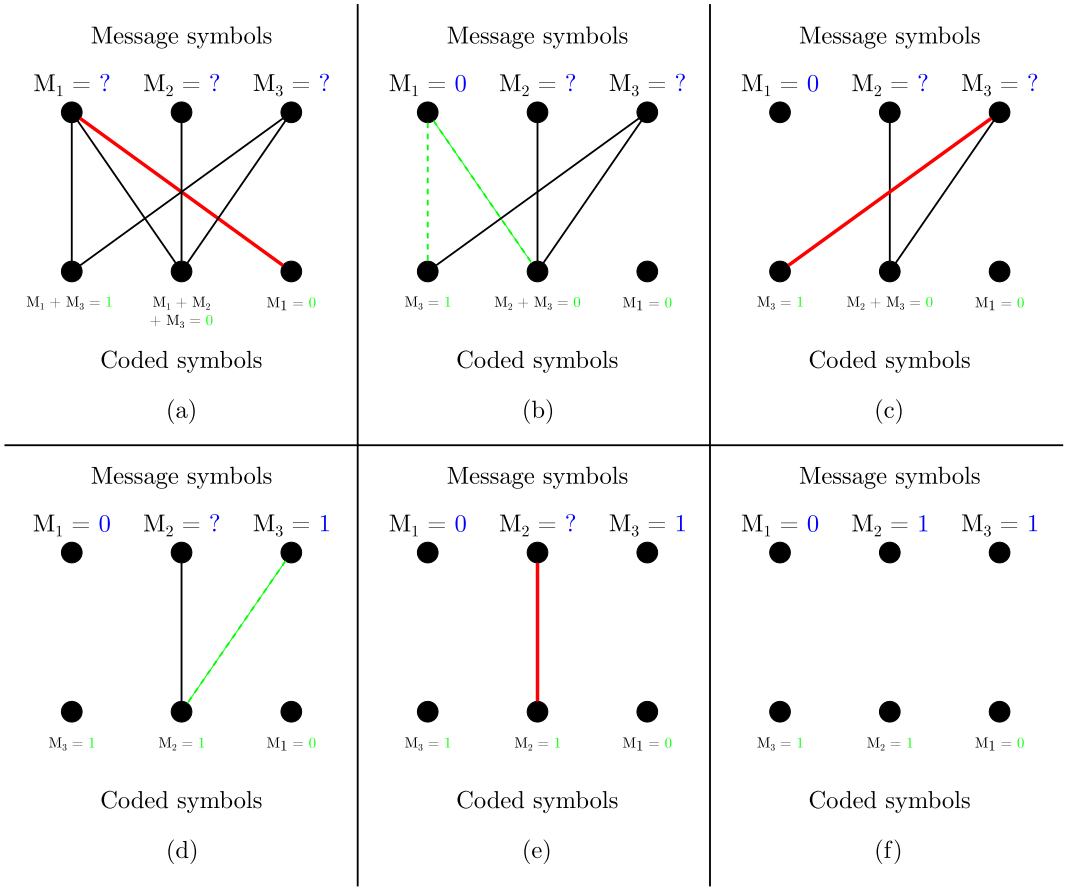


Figure 3.3: Graphic demonstrating the belief propagation algorithm .

3.2.3 Chunking methods and related approaches

Although their main motivations were not to devise a low complexity network code, chunking methods originate from the method detailed in [45]. In a chunking method, the set of messages for broadcast is segmented into *"chunks"* (or sometimes *"generations"*) of constant size, and RLNC is performed on each one. Each packet may be labelled with an additional header to indicate the chunk of which it is a member. Intermediate nodes may perform further coding by forwarding random linear combinations of received messages, but linear combinations must not include messages from multiple chunks. This was suggested as an improvement on the method outlined in [5], in order to overcome delay, packet loss, variability of link quality and capacity, and changing network topology. Notice however, that since the chunks are of constant size, the complexity of decoding each chunk is constant. The decoding complexity of the method therefore increases linearly with increasing data size. Also, since coding is performed over a reduced number of messages, the overhead of communicating the coefficients of each linear combination will be reduced (and indeed constant) [46]. It should be noted that although the authors assert that the method

has linear complexity, the block error probability remains constant, and the overall error probability approaches 1. This is an unfair way to compare computational complexity, as a raptor code can, remarkably, achieve vanishing error probabilities with the same complexity.

Clearly an acceptable approach for unicast connections would be to send each chunk in turn, advancing to the next after an ack transmission had been received from each recipient, signifying that the current chunk had been decoded. The number of acks would be greatly reduced over ARQ in this case, as would the overhead of lost acks (although it is important to note the existence of channels where feedback is not available). However for multicast connections, this method would simply be a return to the issue illustrated in Section 3.1.2, which, however diminished, would still worsen with increasing numbers of receivers. The issue has effectively been "postponed", and the same routing problem the method aims to solve must now be solved for the chunks [46]. Moreover, the process of feedback will incur delay, particularly when the round trip delay is lengthy, and in the case of half duplex devices [47].

The solution presented in [43] is to precode the chunks with a linear complexity code (so that the overall method still has linear decoding complexity, excluding matrix inversion), such that all chunks may be decoded from any large enough subset of the coded chunks. Coded chunks are formed at the transmitter ahead of transmission time. The authors assume that a feedback channel is not available, and at each transmission opportunity a RLNC coded symbol is formed from a randomly selected chunk and transmitted. This approach necessarily means that receivers will occasionally receive symbols formed from chunks which they have already decoded, reducing the bandwidth efficiency of the method. However the redundancy of the precode does mitigate the "curse of the coupon collector" by freeing receivers from waiting for a single last packet, greatly reducing the number of transmissions required until all receivers may decode all chunks. The authors in effect pay a bandwidth penalty for greatly reduced coding complexity.

An extension of this approach suggested in [46] is to allow (or force) the chunks to "overlap", by removing the restriction that chunks must be disjoint. The aim is to reduce the dependency of receivers on a small number of blocks, by allowing decoded blocks (or even individual symbols) to assist the decoding of other blocks. Similar in spirit to the belief propagation decoder, when a receiver decodes a chunk (or some symbols from it), these decoded messages are substituted into the systems corresponding to the other chunks, increasing their rank. One specific method they suggest is to arrange the original message numbers into a rectangular grid, and to

define each row and column to be a chunk. The authors give a bound for such codes, and comment that they have lower complexity than regular chunked codes if the chunk size is constant. The decoding probability of a 2×2 grid code and a chunked code are compared for increasing levels throughput overhead, and the grid code is shown to be superior in all cases, although the authors comment that exact analysis for more complex codes is difficult.

LT codes provide reliable, near capacity achieving communication on erasure channels (both unicast and broadcast), with low complexity encoding and decoding. However, their use is not well suited to network unicast and multicast, as it is non-trivial to re-encode or process the data at intermediate nodes *and* preserve the degree distribution seen at the recipient (indeed, applying an LT encoder at intermediate nodes does not). A generalisation of LT codes for networks known as BATched Sparse codes (BATS) codes are introduced in [47]. Reminiscent of chunked methods which inspired them, the source node of the network produces *batches* of messages in a *rateless* fashion, such that decoding is possible from any sufficiently large set of batches. This is their approach to ending the coupon collector's curse, as no receiver will ever be in need of any specific batch. Each batch is of constant size, containing a total of M coded messages. Reminiscent of LT codes, each batch i is formed by selecting a degree d_i according to a constant degree distribution, and then associating d distinct messages with the new batch. The batch is then formed by taking M random linear combinations of its associated input messages. Intermediate nodes simply broadcast M random linear combinations of the received message packets *from each batch* in turn, but are prohibited from mixing messages from different batches.

So long as the messages received corresponding to each batch i have rank d_i, Gaussian elimination may be performed to recover d_i input messages. Note that as M is constant, this has a constant computational overhead. Similarly to the belief propagation decoder, these recovered input messages may now be substituted into the linear combinations of other batches which are associated with them, which reduces their decoding complexity and may render some of the systems full rank (and therefore decodable). Similarly to overlapping chunked codes, this allows decoded batches to aid the decoding of others.

The authors show that decoding of all but an arbitrary, constant fraction of the input symbols is possible with high probability (for increasing numbers of received symbols) using their decoding method, with a suitably chosen degree distribution. To decrease decoding complexity, some symbol "erasures" by the BATS code are tolerated and recovered by an erasure correcting precode. The method has linear

encoding and decoding complexity, and linear complexity at intermediate nodes. The method also requires only a constant sized buffer at intermediate nodes, as re-encoding is only done over messages from a single batch, and once M coded messages have been transmitted the buffer may be cleared.

Although there has been much research into variations on this idea, all such methods rely on coding over large numbers of messages. This makes them very well suited problems such as the bulk distribution of large files (for instance, computer updates or software), but does nothing to reduce the complexity of decoding for smaller files. The purpose of breaking the messages into chunks is to reduce the complexity of the encoding and decoding, rather than to reduce delay. Indeed, when chunks are pre-coded, it may not be possible to decode chunks until the entire message has been decoded. Moreover, these solutions are not suitable for systems with low temporal relevance, even if they produce data at substantial rates, as buffering of packets is not possible. Chapter 4 will reveal the delay penalty necessary for non-vanishing throughput; the ideal size for disjoint chunks. Chapters 6 and 7 will investigate allcast methods in which each transceiving agent must code over only a single message packet at a time (effectively a chunk of size one).

3.2.4 Sparse Random Linear Network Coding (RLNC)

One method to decrease the decoding complexity of RLNC is to reduce the number of messages included in each linear combination, resulting in *sparse* random linear combinations and *sparse* random decoding matrices. This technique has been adopted in Chapters 6 and 7. This subsection gives an overview other previous work on this technique.

Sparse random square matrices over $\mathbb{F}_q$ with i.i.d entries were studied in [25]. The authors consider the *rank defect* of the matrix: the difference between its rank and the number of columns. Entries are chosen to be zero with fixed probability, and uniformly from the field otherwise. The *density* is defined to be the probability that an entry of the matrix is non-zero. It is shown that if the density is at least $\frac{\log(n)}{n}$, the expected defect is upper-bounded by a constant. Inspired by this article, we have adapted their results considerably to show that matrices which arise from our model are full rank with high probability. Using Gaussian Elimination to invert an $n \times n$ matrix where entries are chosen uniformly from a finite field has complexity $O(n^3)$, although an improved version of Gaussian elimination algorithm is introduced in [48] which achieves this with complexity $O(n^3/\log(n))$. However, an efficient algorithm for solving sparse systems of linear equations over finite fields is given in [49] which can invert matrices with density $\frac{\log(n)}{n}$ with complexity $O(n^2 \log(n))$.

The computational expense of sparse RLNC is explored through experimentation in [50], where it is shown that the decoding rate of a particular decoder can be decreased by an order of magnitude by decreasing the density, without considerably reducing the throughput of the method. The use of sparse RLNC codes for broadcast transmissions over erasure channels is considered in [51]. The authors provide an accurate approximation for the probability of all users being able to decode every message.

Tunable codes, in which the density of linear combinations evolves over time, were first proposed in [52]. The authors show that a modified Gaussian elimination algorithm may decode messages with constant density in linear time, but that excessively many of them are required to form a linearly independent subset. The authors suggest switching to transmitting dense linear combinations for the last 10% of transmission opportunities. Although this does not reduce the complexity order overall, it was shown to reduce the computational expense of decoding in simulations. This idea is refined in [53], which shows that the probability of a message being *innovative* (linearly independent with the messages a receiver has already collected) reduces with the number of messages already collected, and increases with density. This makes rigorous the assertion from their previous paper; that the last innovative packets are the hardest to obtain. The authors suggest using feedback to inform the transmitter once the number of received packets exceeds a sequence of thresholds, and for the transmitter to increase the density on receipt of feedback. This aims to achieve balance in a tradeoff of a desire to reduce both complexity and the overhead of feedback.

The authors of [54] consider a problem related to distributed storage. Each of k nodes generate a packet of data, and it is desired for $n > k$ storage nodes to store a single packet each, in such a way that all k data packets may be recovered from any k storage packets. The authors introduce a RLNC inspired method, in which each storage packet is the sum of d data packets. The authors show that any k storage packets are linearly independent with high probability (for increasing field sizes) if $d = O(\log(n))$, and that this is the optimal choice for d. The density of these matrices is similar those in Chapters 6 and 7; although their density is fixed, it matches the expected density of these matrices. The analysis in these chapters will, however, consider binary fields, in exchange for extra transmissions.

3.3 Broadcasting over erasure channels

Chapter 4 considers the problem of a single transmitter, broadcasting an infinite stream of messages to a fixed number of receivers, over a broadcast erasure channel. The chapter quantifies the tradeoff between throughput and delay for fountain coding, and a compares the performance of the method with ARQ for these quantities. This section summarises previous work related to Chapter 4.

3.3.1 Throughput-delay trade-off

The Fundamental Theorem for a Discrete Channel With Noise from Claude Shannon's famous 1948 paper [3] states that for any discrete channel with capacity C, a coding system exists which can transmit information over the channel at rate R with arbitrarily small error probability. Shannon's proof considers randomly selected codes with rate $R < C$ with blocklength N (i.e where encoding and decoding is performed over N symbols), and the average block decoding error $\bar{P}_e$ is shown to decrease exponentially in N. Since there must exist at least one code with block decoding error probability P_e at most $\bar{P}_e$, this proves the existence of codes with arbitrarily small P_e. This is in contrast to the previous belief that reliability must come at the cost of reduced throughput, as is the case with repetition coding, which achieves vanishing throughput as $P_e \longrightarrow 0$.

However, arbitrary P_e is achieved by increasing N, necessarily incurring a delay of N time-steps on the system before decoding may commence. Shannon's proof does show the relationship between $C - R$ and the rate of decay of $\bar{P}_e$, although the proof given by Gallager in [55] for Discrete Memoryless channels (here known as the Noisy Channel coding theorem) makes this explicit. He shows $\bar{P}_e < \exp(-N \cdot E_r(R))$, where E_r is a convex, non-negative, non-decreasing function on $[0, C)$. Moreover, $E_r(C) = 0$. For tighter bounds and a review of similar work, see [56]. The result highlights a trade-off between the block error probability, the throughput achieved and the delay incurred; the error rate decays exponentially in N, the rate of decay increases if throughput is sacrificed, and throughput may be traded for delay for fixed error rates.

This is a useful concept in practice. For instance, an end user downloading computer updates may not care how long it takes to download them, but efficient usage of available bandwidth may be paramount for operational reasons. Conversely, the delay which applications such as real-time multimedia or voice/video calls may tolerate could be very small indeed, and it may be preferred to accept lower bandwidth or tolerate higher error rates in exchange for increased throughput.

The throughput-delay trade-off of HARQ over Additive White Gaussian Noise (AWGN) channels is investigated in [57]. The authors assumed that a single transmitter wishes to multicast a message of b bits to K receivers, over channels with i.i.d gain which is constant for the duration of a packet. Feedback channels are assumed to be error and delay free. A randomised coding argument is used in each case to prove the existence of codes which meet their bounds. In the case of standard ARQ, their model reduces to a broadcast erasure channel with perfect feedback, for which the authors show that the average delay and per-user throughput scales as $\log(k)$ and $\frac{1}{\log(k)}$ respectively (consistent with the our asymptotic bounds in Section 4.2). In the case of *incremental redundancy*, in which redundant packets are transmitted in place of re-transmissions (and concatenated before decoding), it is shown that any fraction of throughput is achievable in exchange for a linear delay penalty.

3.3.2 Related Work

The model and problems considered in Chapter 4 and extensions thereof have been well studied in the literature. The main contribution made in the chapter is the asymptotic bound provided in Section 4.2. Whilst the model omits many real world features, greater modelling assumptions have arguably resulted in greater utility.

The authors of [58] show that RLNC is throughput optimal for broadcast on our model. The authors of [59] consider a system in which N_i RLNC messages are broadcast in each iteration i over an identical channel model to ours, before receiving feedback from each receiver detailing the number of linearly independent messages required. A bound is derived on the number of iterations required for all users to receive the message, with probability exceeding a specified threshold. A computationally expensive algorithm is also presented for minimising the expected completion time by optimally choosing N_i, as well as some computationally feasible heuristics.

The delay distribution of RLNC for broadcasting over broadcast erasure channels is derived in [60]. The authors define the *degrees of freedom* held by a receiver to be the rank of the set of linear combinations they have received. The authors observe that the degrees of freedom held by all receivers at each time-step behaves as a Markov chain. In the case of one and two receivers, it is shown that the number of possible transitions is small enough that the transition matrix may feasibly be computed. The probability distribution for the delay may then be computed by computing powers of the matrix. Whilst this is already computationally expensive, the authors note that for larger numbers of receivers, this approach is computationally infeasible, and suggest a brute force approach instead.

The analysis is extended to time-varying channel erasure probabilities in [38]. The authors derive an exact expression for the expected number of transmissions required for successful reception of a fixed length message by all broadcast receivers, when RLNC is employed. A comparison of numerical values of this expectation, as well as numerical and simulated expected completion times for a number of other scheduling algorithms detailed in the paper, shows that RLNC is superior in all cases. Making the simplifying assumption of time invariant channels, this chapter contributes an approximate method for determining the number of packets which must be transmitted in order for every user to receive the message with high probability, giving a better method for determining how many coded packets to broadcast, which takes the variability of this requirement into account.

Paper [61] considers RLNC applied to a time invariant broadcast erasure channel, where the transmitter is linked to each receiver by an independent erasure channel, with distinct erasure probabilities. The author derives a lower bound for the number of transmissions required for successful reception of a fixed length message by all broadcast receivers, and shows this to be close to the true expressions for practical system parameters in simulations. The same model is considered by [62], where the expected delay for RLNC and ARQ are derived exactly, and the delay of RLNC is shown to be lower asymptotically than ARQ for increasing C. The authors extend their results in [63], including bounds on the moments of the expected delay, and prove that the expected delay per packet is monotone decreasing in the buffer size.

Our model is extended in [64] to eliminate the assumption that the transmitter always has messages to transmit at each time-step, by assuming that messages arrive at the transmitter according to a Bernoulli process, and are enqueued for coding and transmission in a buffer of infinite size. Our model is retained as a special case. The authors assume the transmitter applies a fountain code over blocks of k messages at a time (once this many have arrived in the buffer), and derive exactly the distribution of the number of time-steps required by both fountain coding and ARQ before all n receivers may decode all k messages. Simpler, more powerful asymptotic results are derived in this chapter.

The results of the papers introduced above are cumbersome; indeed, many must be evaluated (or approximated) by a computer. A simpler, more powerful approximation is derived in Section 4.2, which is the result of making additional assumptions compared with some of these articles.

The papers above considers the buffer size and number of receivers mostly in isolation, and at best comment that larger buffer sizes improve performance. An asymptotic throughput-delay analysis is given in [65] for a model identical to that

in section 4.1. The analysis is in extended in [66] to a model in which erasures are correlated over time. The communications links evolve according to a two-step Markov chain, in which the states represent erasure of the message at that time-step, and the error-free communication of the transmitted message. The paper considers the case in which the buffer size is allowed to grow as a function of the number of receivers, and the broadcast throughput, which they define as the limit as time $t \longrightarrow \infty$ of the average number of successful messages transmitted to all per time-step. The authors discover a phase transition; if the buffer size may grow faster than $\log(n)$, then the broadcast throughput will approach capacity with the number of receivers n. If the buffer size may grow as $\Theta(\log(n))$, then the lim inf of the broadcast capacity is equal to a constant fraction of the channel capacity. If the buffer-size grows sub-logarithmically, then the broadcast throughput approaches zero.

Whilst the authors do find approximations for the mean and variance of the delay, this analysis assumes the buffer size grows super-logarithmically. Section 4.2 provides a full throughput-delay tradeoff for the case in which the buffer size grows logarithmically, showing the relationship between the scaling of the buffer size and the logarithmic scaling of the delay explicitly. The approximations of the expected delay in these two articles are complemented by asymptotic upper bounds.

The authors of [67] study the use of RLNC to broadcast messages arriving in a queue over a time-varying broadcast erasure channel to n receivers. The authors show that if the buffer size is allowed to grow as a fixed (specified) multiple of $\log(n)$, then the expected number of time-steps before all receivers may decode is upper-bounded by a multiple of $\log(n)$. The work of Section 4.2 generalises this (with the advantage of assuming perfect fountain coding) by relating the buffer size $\alpha \log(n)$ with the delay $\beta_\alpha \log(n)$, and making the relationship between α and β_α explicit. This work is also complementary to this chapter, bounding the expected delay as opposed asymptotic bound on the delay itself.

Other papers study a similar channel model, but further assume the availability of perfect (error and delay free) feedback channels, from each receiver to each transmitter. Paper [68] considers an "online" style method, in which packets are included for transmission (potentially) as soon as they arrive, rather than waiting for the current buffer of messages to be decoded. Each receiver informs the transmitter once it has *seen* a given message, i.e when it is able to compute a linear combination involving the packet and only newer packets. Packets which have been seen can be decoded as soon as all newer packets have been decoded, and it will make no difference to these receivers if these messages are included in linear combinations they receive, so the transmitter drops packets from its buffer once they have been seen

by all receivers. The transmitter constructs linear combinations at each timestep such that, on successful reception, each receiver will see its next unseen packet. The authors show that the expected number of messages which have not been seen by all receivers at each timestep is constant, given a Bernoulli arrival process. A similar model is studied by [69], which studies *instantly decodable* network codes (introduced in Section 3.4). Their method uses feedback to allow a linear combination to be formed which can be decoded instantly by a large number of receivers. Whilst it is unfair to say that feedback is wasteful in these situations, in applications it will not come for free, as these papers assume. Arguably these methods are therefore trading throughput (especially where uplink and downlink share the same medium) for reduced coding complexity.

3.4 Allcast: Multiple simultaneous network broadcasts

Chapters 5, 6 and 7 consider the allcast problem, in which a fixed number of agents must each broadcast a message to all others. The latter two chapters introduce RLNC algorithms, whilst Chapter 5 gives an uncoded algorithm for comparison. The issue of multiple broadcasts, and sharing of data amongst a group of agents, has several applications. In vehicular networks, this form of communication can be used to share sensor readings (such as the road surface and traffic conditions) [10], as well as to control and optimise vehicular speeds [7]. Other applications of allcast communication include group key exchange algorithms, in which a group of agents wish to agree on a common secret key over a public network [70].

The application of RLNC to allcast was first proposed in [71], in the form of a *gossip algorithm*. The authors in this case adopt a *random phone call* model, in which agents in each round select another single node, and transmit a coded message to them. They show that their method allows faster dissemination than uncoded methods. The work of [72] analyses RLNC gossip in great generality, and whilst their results are applicable to our model, the resulting bounds are not tight enough to be useful.

The authors of [73] consider allcast over complete undirected graphs, where the capacity of each edge is chosen i.i.d at random. The authors analyse the capacity region of their model, and present an uncoded push-pull allcast method, which they show to be asymptotically optimal. The authors do not however consider broadcast channels, and assume that different messages may be sent to each adjacent node (as they are modelling wired networks).

In [74], an RLNC allcast system is analysed, on graphs where edges denote era-

sure channels, and the medium is broadcast in nature. A multiple access system is in operation, restricting nodes to broadcast one at a time, and ensuring that nodes gain channel access with equal probability (modelling Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA)). The authors show that the average stopping time (and total number of transmissions) for the complete graph is $O(n)$. We complement with analysis of a model and system where erasures are correlated over time.

The allcast problem has been studied in conjunction with the multiple access problem for wireless broadcast channels. When a pair of nodes broadcast, it is assumed that any common neighbour to them will be unable to receive either message; a phenomenon known as a *collision*. Scheduling solutions to avoid collisions are considered is [75]. Since finding an optimal schedule is an NP-hard problem, the authors introduce three approximate algorithms. One method finds a schedule which prevents collisions. The authors assume the nodes are randomly spaced, and that nodes within a given range may communicate error free (in the absence of collisions). One method decomposes the graph into a spanning tree, so that leaves are assigned parent nodes which forward all their messages for them. The "gather scatter" method involves decomposing the graph into a tree, and scheduling alternative layers to transmit in each time-step in order to relay all messages to a chosen root node. Messages are then distributed from there. The authors claim this to work best in practice according to their experiments. All of these algorithms have polynomial complexity. The authors of [76] consider a similar model with arbitrarily placed nodes. They introduce a method in which the graph nodes are grouped according to their breadth first search tree, starting at the graph center, and nodes are chosen to interconnect layers. The messages are collected by using a 3-intermittent schedule of layers, so that nodes spaced fewer than two layers apart never transmit concurrently. A schedule is then formed for distributing the messages, using intermediate nodes to forward messages to leaves of the tree. Our model does not consider the multiple access problem as these papers do, and we leave this as further work.

A related method to those studied in Chapters 6 and 7 is COPE [77], and a more general class of techniques known as *Instantly Decodable Network Codes (IDNC)* [78]. Noting that when using RLNC, individual messages cannot in general be decoded before full decoding is possible, these methods instead ensure that each coded message is decodable by its recipients *immediately*. These codes achieve this using a system of feedback (possibly aided by channel/erasure estimation [77]), so that agents can keep track of which of their neighbours have received *and decoded* each message. At each timestep, each agent will form a subset of the messages they

have to transmit, such that a large number of agents will have received all but one of the messages, and then broadcast the sum of them. Each of these such agents will be able to decode the message instantly on reception. This greatly reduces the computational complexity of the decoding, compared to the Gaussian elimination algorithm required for RLNC. Any messages received which cannot be decoded by an agent will be discarded, and not used in any way to aid decoding, which is wasteful but saves the requirement to buffer coded messages.

Paper [78] considers a graph, in which each vertex V_{ij} corresponds to a message j which has not been received by a particular agent i. Edges exist between vertices exactly when both corresponding agents have not received the corresponding message, or one has received it but the other has not. The authors show that finding the optimal set of messages to include in a linear combination is equivalent to finding a maximum clique in the graph. Unfortunately, whilst optimal in terms of delay on packet decoding, this encoding algorithm is NP-hard. Generally, polynomial time heuristics are studied instead [78]. The methods typically require more transmission time-steps than RLNC before decoding is possible, and this may be traded off in exchange for lower overall delay [69].

An IDNC system which achieves multiple multicasts over erasure channels is introduced in [79]. The authors assume that a fixed number of transmitters each wish to communicate (possibly non-disjoint) sets of messages to a fixed number of receivers. The transmitters are assumed not to have direct communication links with each-other, and to have error-free bi-directional links with a control node. The feedback channels for the receivers are also assumed to be error-free, whilst erasures of packets arriving at the receivers are assumed to occur with fixed probability (for each receiver) and be i.i.d over time and space. After all uncoded messages have been sent, the control node schedules subsequent broadcast transmissions, using its knowledge of which receivers have received which messages, obtained from feedback. This model does not extend to slow fading as studied in Chapters 5, 6 and 7. Moreover, the method is not distributed, and would require infrastructure or some form of negotiation process to elect a control node. The requirement of a direct link between transmitters and the control node prevents application of this method to allcast in the presence of shadowing.

The majority of research on the topic of IDNC deals with implementation details, and is much more applied than this thesis. Analysis is mostly done through simulations and practical experiments, in contrast to this thesis which mostly concerns scalability to large networks, and asymptotic bounds to give insight on how the algorithms scale with network size. This not only makes comparison of the re-

sults of these papers difficult, but unfair, as the models/experiments are designed to capture different physical phenomena.

This page is left intentionally blank

Chapter 4

Throughput-delay trade-offs for the broadcast erasure channel

This chapter considers the problem of a single transmitter, broadcasting an infinite stream of messages to a fixed number of receivers, over a broadcast erasure channel. The transmitter may employ one of two error correction methods, Automatic Repeat Request (ARQ) and fountain coding. The tradeoff between throughput and delay will be quantified for fountain coding, and a comparison made with ARQ for these quantities.

The model introduced in this chapter is applicable to the engineering problems introduced in Chapter 1. For instance, the model is relevant to a single vehicle broadcasting data to those surrounding it, in channels exhibiting packet loss due to fast fading. Or this model could consider a single server pushing data to subscribers over Internet Protocol (IP) multicast, which experience packet loss at the network layer (due to error, congestion, or otherwise).

The content of this chapter is an extended and adapted version of work published in [23], and is joint work with Ayalvadi Ganesh and Robert Piechocki.

4.1 System Model

Consider a single agent, known as the transmitter, which possesses an infinite stream of messages. No assumptions about the format or content of these messages will be made in this chapter, but it will be assumed that the transmission of each message will take the same amount of time, and be subject to loss at the same rate. In practice, this likely restricts the messages to be of identical length.

These systems shall be indexed by n, and in the n^{th} system, the agent will broadcast the stream of messages to n receivers, indexed by i. Every time the

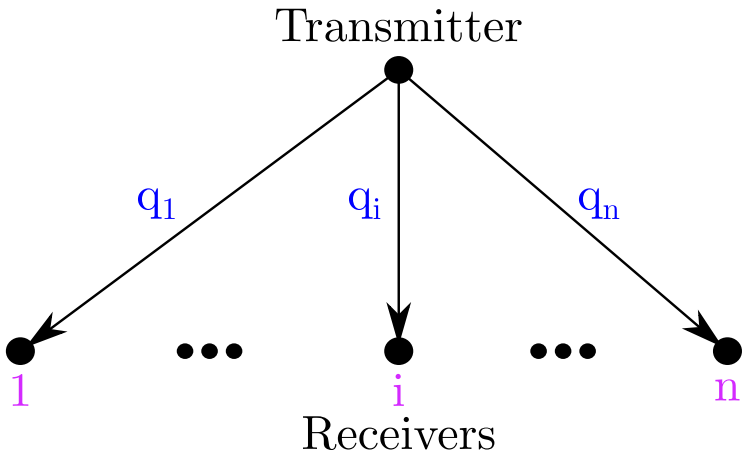


Figure 4.1: Graphic illustrating the broadcast erasure channel model.

transmitter transmits a message, each receiver i either receives the same message error free, or receives nothing independently with probability q_i (a phenomenon known as an *erasure*). In addition, erasures are independent across time. Reception is assumed to be synchronised at the receivers, occurring at discrete time intervals (referred to herein as time-steps), long enough for transmission of any message. This channel is known as the *broadcast erasure channel*. Figure 4.1 gives an illustration of the broadcast erasure channel for convenience. Whilst the graphic has been produced from scratch, as this is a well studied model and a simple illustration, naturally it bears similarity to many others in the literature (e.g [63]).

In addition, it is assumed that the erasure probabilities are chosen i.i.d at random before communications commence, according to an arbitrary probability distribution ψ supported on $[q_{\min}, q_{\max}] \subset (0, 1]$. For convenience, define $\Psi(x) = \mathbb{P}(q_i < x)$ to be the Cumulative Distribution Function (CDF) of the erasure probabilities. Note that this generalises the case in which all links have the same erasure probability (as studied in Section 4.3), which may be obtained by taking ψ to be the distribution of a constant random variable. This does not include the case in which the links have arbitrary and distinct erasure probabilities (as studied in [62, 63]), however in practice this model is arguably more realistic. Unless the receivers are known ahead of time, as opposed to ad-hoc communication, it is impossible to know the exact erasure probabilities, but more reasonable to assume their distribution may be known.

The transmitter is permitted to use one of two error correction schemes (which will both be investigated in the next section), about which generous assumptions will be made. In the case of ARQ, it is assumed that an ideal, delay and error free feedback channel is available from each receiver back to the transmitter. That

is, at each time-step, the transmitter is aware of any erasures which occurred at the previous time-step. It is well known that this assumption itself is generous, and in practice the delay will cause a considerable reduction in throughput [15]. The transmitter will broadcast one message at a time (and it is assumed that a message is always available to transmit at each time-step). It is assumed that the transmitter has sufficient memory to keep a log of which messages have received by each receiver, and the computational power available to update this at each time-step. The transmitter will repeatedly transmit the same message until every receiver has received it. This notably ignores the famous feedback implosion problem (see Section 3.1.1 for a full discussion of the issues with ARQ). It should be clear to the reader that despite these generous assumptions, the number of redundant messages received by each receiver is extremely wasteful of the link capacity [28], and it is easy to show that this alone will cause the rate of throughput achieved by ARQ to vanish as $n \longrightarrow \infty$.

The transmitter may instead employ *perfect* fountain coding, where the word "perfect" symbolises the idealising assumptions made by this model. The transmitter will buffer k messages at a time, and it is further assumed that it has enough memory to do this. Once the buffer is full, the transmitter will broadcast coded messages formed from messages in the buffer, until all receivers can decode every message in the buffer. It is assumed that the transmitter either has possession of an infinite number of messages, or that it receives them at a rate sufficiently rapid (and has sufficient buffering space), so that a full buffer of messages is always on hand to be transmitted after the previous one has been fully disseminated. It is assumed that decoding is possible exactly when a node receives *exactly* k coded messages, regardless of which messages are received. This assumption is somewhat generous. Consider the case of Random Linear Network Coding (RLNC), in which coded messages are random linear combinations of the buffered messages, with coefficients chosen uniformly at random from $\mathbb{F}_r$. Then by [38], the expected number of received messages required to obtain a linearly independent k-subset is upper-bounded by $k\frac{r}{r-1}$. It is clear that this approaches k with r, but is close even for moderate field sizes, justifying the assumption in this case. However, some forms of fountain coding such as Raptor codes choose to tolerate a penalty of unnecessary redundancy to reduce decoding complexity, and by design require more than k coded messages to be received to decode k [44]. The analysis in this section does not take this into account. Another consequence of this assumption is that partial decoding is never possible, despite the existence of systematic codes such as Raptor codes. Hence, a delay is incurred for all messages, and this is the topic of interest in this chapter.

The maximum rate achievable by any coding scheme is $1/(1-\max_{i=1}^{n} q_i)$, as that is the channel capacity to the worst-off receiver. It is well known that RLNC can achieve this rate [58]. In contrast, ARQ requires $X_i \sim Geom(1-q_i)$ transmissions of a message until receiver i gets it, and $\max_{i=1}^{n} X_i$ transmissions for all receivers to get it. Lemma 2.9 shows that this number scales as $\log n$. Hence, the throughput of ARQ scales as $1/\log n$, which vanishes as the number of receivers increases to infinity. It is less obvious how the delay of these schemes scale with k and n, which is the topic of the next section.

To end this section, we briefly explain the choice of ARQ as a baseline solution. Alternative candidates include Hybrid Automatic Repeat reQuest (HARQ) (and other methods based on fixed-rate coding), such as that studied by [80], in which additional transmissions of parity packets are made as needed. This method can perform no better than ideal fountain coding under this model, as after large numbers of erasures, the transmitter would be force to repeat messages which will be unhelpful to some receivers. Ideal fountain coding may be a reasonable model of this method however if proportion of successful message receptions is above the coding rate.

HARQ can also be implemented *within* a packet, as is the case for the data links in Fifth Generation-New Radio (5G-NR) [81], where packets received in error are corrected using additional parity transmissions. Whilst these transmissions are smaller, they still only aid the decoding of a single message. If we assume that with constant probability q a receiver experiences enough symbol errors to prevent decoding with *all* of the parity symbols of the underlying code, then the probability that *none* of the receivers experience this decays exponentially fast as $(1-q)^n$. In fact, with high probability, a fixed fraction nq of the receivers will experience this. There is no fixed coding rate which can prevent this, which will cause the transmitter to resort to re-transmissions in every round. In effect, HARQ will reduce to ARQ with fixed rate coding for a large enough number of receivers, unless the coding rate is allowed to decay to zero, to allow q to decrease with n. The issue is that even rare events will occur for for at least *one* receiver if there are enough of them. The next section will show that the same is not true for their *average* behaviour over a long enough period of time.

Finally there are methods which use feedback to allow individual messages to be decoded before the rest of the buffer [68] (perhaps even instantly on receipt of coded messages [69]). Unfortunately this does necessitate the use of feedback (as do the methods above), perhaps in a useful way, but at a cost. Suppose that the medium is shared between downlink and feedback, with a perfect Medium Access Control (MAC) solution, such that each receiver informs the transmitter whether

or not they have received each message after transmission. If the message is of constant size, and all acknowledgements are of equal and constant size, then it is clear that the throughput achieved will decay as $\frac{1}{n}$. The next section will show the delay penalty to be paid for non-vanishing throughput as the number of receivers increases. Even if feedback is deployed (to guarantee perfect reception before the receiver moves to its next buffer), then at worst each receiver must acknowledge just the entire buffer. Although this would give throughput decaying as $\frac{\log(n)}{n}$, a substantial improvement could be made by forcing receivers to wait until the number of transmissions (determined by headers) exceeds the asymptotic bounds in the next section, before sending *negative* acknowledgements as necessary. The probability of exceeding these bounds decays as $n^{-\epsilon}$, and so the expected number of acknowledgements will decay slowly as $\frac{\log(n)}{n^{\epsilon}}$, whilst the number of payload messages will grow unbounded.

4.2 Throughput-delay trade-offs

Clearly under this model of fountain coding, if the buffer size $k = 1$, the model reduces to one of ARQ. It is clear therefore that in order to gain any advantage from using fountain coding, the buffer size must be increased. The research question of this section is by just how much, and more precisely, to find an asymptotic regime under which fountain coding may provide constant throughput for increasing numbers of receivers. It will turn out that in order to achieve non-vanishing throughput, the buffer size must increase logarithmically. The main result in this section is the following theorem, which shows that fountain coding can achieve the full range of possible rates, from zero up to channel capacity, with latency that is logarithmic in the number of receivers (albeit with a constant prefactor that depends on the rate).

Theorem 4.1. *Suppose the buffer size k is allowed to grow with n, which will herein be denoted by k_n to make its dependence on n clear. Suppose the transmitter employs perfect fountain coding over blocks of k_n packets. Denote by $T = T(n, k_n)$ the random number of time-steps until all n receivers have decoded all k_n packets in the message. We have the following:*

$$\text{If } \frac{k_n}{\log n} \to \alpha \geq 0, \text{ then } \frac{T(n,k_n)}{\log n} \xrightarrow{p} \beta_\alpha,$$

where $\beta_\alpha := \inf\Big\{\beta > \frac{\alpha}{1-q_{\max}} : \beta D\Big(\frac{\alpha}{\beta}; 1-q_{\max}\Big) > 1\Big\}$ *and* $\xrightarrow{p}$ *denotes convergence in probability. Moreover, the function* $\beta \mapsto \beta D\big(\frac{\alpha}{\beta}; 1-q_{\max}\big)$ *is non-decreasing on* $\big[\alpha/(1-q_{\max}), \infty\big)$.

It is easy to see from properties of the relative entropy function that the set over which the infimum in the definition of β_α is taken is non-empty for all $\alpha \geq 0$, and that the infimum is attained; consequently, β_α is finite.

Proof. By the assumption of perfect fountain coding, receiver i has decoded the message by time ℓ if it has received at least k_n packets by this time, i.e., suffered no more than $\ell - k_n$ erasures. The number of erasures experienced by receiver i in ℓ time slots is binomially distributed with parameters (ℓ, q_i). Hence, letting T_i denote the random time at which receiver i decodes the message, we have

$$\mathbb{P}(T_i > \ell) = \mathbb{P}\big(Bin(\ell, 1-q_i) < k_n\big) \leq \mathbb{P}\big(Bin(\ell, 1-q_{\max}) < k_n\big),$$

where the final inequality follows by Lemma 2.11. As

$$T(n, k_n) = \max_{i=1}^{n} T_i,$$

it follows from the union bound that

$$\mathbb{P}(T(n,k_n) > \ell) \leq n\mathbb{P}\big(Bin(\ell, 1-q_{\max}) < k_n\big).$$

Hence, by Lemma 2.10, we have for $\ell > \frac{k_n}{1-q_{\max}}$ that

$$\log \mathbb{P}\big(T(n,k_n) > \ell\big) \leq \log n - \ell D\Big(\frac{k_n}{\ell}; 1-q_{\max}\Big). \tag{4.1}$$

Fix $\epsilon > 0$, $\beta > (1+\epsilon)\beta_\alpha$ and take $\ell_n = \lceil \beta \log n \rceil$. By the assumption that $k_n/\log n$ tends to α, $\frac{k_n}{l_n} < (1+\epsilon)\frac{\alpha}{\beta_\alpha}$ provided n is sufficiently large. By the monotonicity of $\beta \mapsto D\big(\frac{\alpha}{\beta}; 1-q_{\max}\big)$ on $[\alpha/(1-q_{\max}), \infty)$ (by Lemma 2.4), for such n we have

$$\begin{aligned}\ell_n D\Big(\frac{k_n}{\ell_n}; 1-q_{\max}\Big) &\geq (1+\epsilon)\log n \beta_\alpha D\Big(\frac{\alpha}{\beta_\alpha}; 1-q_{\max}\Big) \\ &\geq (1+\epsilon)\log n,\end{aligned}$$

where the last equality holds by the definition of β_α and the continuity of $D(\cdot; 1-q_{\max})$. Substituting the above inequality into (4.1), we obtain

$$\limsup_{n\to\infty} \mathbb{P}\big(T(n,k_n) > (1+\epsilon)\beta_\alpha \log n\big) \leq \limsup_{n\to\infty} n^{-\epsilon} \;=\; 0. \tag{4.2}$$

We need a corresponding lower bound on $T(n,k_n)$. Clearly $\alpha(1-q_{\max}) \notin \Big\{\beta > \alpha(1-q_{\max}) : \beta D\Big(\frac{\alpha}{\beta}; 1-q_{\max}\Big) > 1\Big\}$ since $D(1-q_{\max}; 1-q_{\max}) = 0$, and is not a limit point of the set by continuity of $D(\cdot; 1-q_{\max})$. Hence, $\beta_\alpha \neq \alpha(1-q_{\max})$. Hence we may let $\epsilon > 0$ such that $\frac{\alpha}{(1-\epsilon)^2\beta_\alpha} < 1 - q_{\max}$. Let $\delta = \inf\Big\{\delta^* \in (0, q_{\max} - q_{\min}) : \sqrt{1-\epsilon} D\Big(\frac{\alpha}{(1-\epsilon)^2\beta_\alpha}; 1-q_{\max}+\delta^*\Big) \leq D\Big(\frac{\alpha}{(1-\epsilon)^2\beta_\alpha}; 1-q_{\max}\Big)\Big\}$. Clearly by monotonicity (Lemma 2.4) the set is non-empty and hence δ is finite and positive.

Fix i, and assume $q_i \in [q_{\max} - \delta, q_{\max}]$. Observe that T_i, the number of transmissions until receiver i gets k_n packets, satisfies

$$\begin{aligned}\mathbb{P}(T_i \leq \ell) &= \mathbb{P}\big(Bin(\ell, 1-q_i) > k_n\big) \\ &\leq \mathbb{P}\big(Bin(\ell, 1-q_{\max}+\delta) > k_n\big) \\ &\leq 1 - \binom{\ell}{k_n}(1-q_{\max})^{k_n} q_{\max}^{\ell-k_n},\end{aligned}$$

where the first inequality follows by Lemma 2.11. Now, for k_n and ℓ tending to infinity, we have by Stirling's formula that

$$\begin{aligned}\binom{\ell}{k_n} &\sim \Big(\frac{\ell}{k_n}\Big)^{k_n}\Big(\frac{\ell}{\ell-k_n}\Big)^{\ell-k_n}\sqrt{\frac{\ell}{2\pi k_n(\ell-k_n)}} \\ &\geq \frac{1}{\sqrt{2\pi\ell}}\Big(\frac{\ell}{k_n}\Big)^{k_n}\Big(\frac{\ell}{\ell-k_n}\Big)^{\ell-k_n},\end{aligned}$$

where, for two sequences x_n and y_n, we write $x_n \sim y_n$ to denote that they are asymptotically equivalent, i.e., x_n/y_n tends to 1 as n tends to infinity. Thus, we obtain for arbitrary $\epsilon > 0$ and all ℓ and k_n sufficiently large that

$$\mathbb{P}(T_i \leq \ell) \leq 1 - \frac{1-\epsilon}{\sqrt{2\pi\ell}}\exp\Big(-\ell D\Big(\frac{k_n}{\ell}; 1-q_{\max}+\delta\Big)\Big). \tag{4.3}$$

Let $\ell_n = (1-\epsilon)\beta_\alpha \log(n)$. Since $\frac{k_n}{\log(n)} \longrightarrow \alpha$, for large enough n we have $\frac{k_n}{\ell_n} \leq \frac{\alpha}{(1-\epsilon)^2\beta_\alpha}$, and consequently

$$\begin{aligned}\mathbb{P}(T_i \leq \ell_n) &\leq 1 - \tfrac{\lambda}{\sqrt{\log(n)}}\exp\Big(-(1-\epsilon)\beta_\alpha\log(n) D\big(\tfrac{\alpha}{(1-\epsilon)^2\beta_\alpha}; 1-q_{\max}+\delta\big)\Big) \\ &\leq 1 - \tfrac{\lambda}{\sqrt{\log(n)}}\exp\Big(-\sqrt{1-\epsilon}\beta_\alpha\log(n) D\big(\tfrac{\alpha}{(1-\epsilon)^2\beta_\alpha}; 1-q_{\max}\big)\Big) \\ &\leq 1 - \tfrac{\lambda}{\sqrt{\log(n)}}\exp\Big(-\sqrt{1-\epsilon}\beta_\alpha\log(n) D\big(\tfrac{\alpha}{\beta_\alpha}; 1-q_{\max}\big)\Big) \\ &\leq 1 - \tfrac{\lambda}{\sqrt{\log(n)}}\exp\big(-\sqrt{1-\epsilon}\log(n)\big) \\ &\leq 1 - \frac{\lambda n^{-\sqrt{1-\epsilon}}}{\sqrt{\log(n)}},\end{aligned} \tag{4.4}$$

for some constant $\lambda > 0$. Let $B = \big\{i \in \{1,\ldots,n\} : q_i \in [q_{\max}-\delta, q_{\max}]\big\}$ denote the set of receivers for which the erasure probability is within δ of its maximum $q_{\max}$. Let $\mu = \frac{|B|}{n}$ and note that $\mu n = |B| > (1-\epsilon)(1-\Psi(q_{\max}-\delta))n$ with high probability. The total number of transmissions until all n receivers can decode the k_n message packets is given by $T(n,k_n) = \max_{i=1}^n T_i \geq \max_{i\in B} T_i$. Moreover, the random variables T_i are mutually independent by the assumption that erasures on channels to distinct receivers are mutually independent. Hence, if $i^* \in B$,

$$\begin{aligned}\mathbb{P}\big(T(n,k_n)\leq \ell\big) &\leq \mathbb{P}(\forall i\in B: T_i\leq \ell)\\ &\leq \mathbb{P}(T_{i^*}\leq \ell)^{\mu n}\mathbb{P}(|B|\geq \beta n)+\mathbb{P}(|B|<\beta n)\\ &\leq \mathbb{P}(T_{i^*}\leq \ell)^{\mu n}+\exp(-\theta n),\end{aligned}$$

for some $\theta\in\mathbb{R}^+$. Substituting (4.4) into the above, we obtain

$$\begin{aligned}\mathbb{P}\big(T(n,k_n)<(1-\epsilon)\beta_\alpha\log n\big) &\leq \Big(1-\tfrac{\lambda n^{-\sqrt{1-\epsilon}}}{\sqrt{\log(n)}}\Big)^{\mu n}+e^{-nk}\\ &\leq \exp\Big(-\tfrac{\lambda\mu n^{1-\sqrt{1-\epsilon}}}{\sqrt{\log(n)}}\Big)+e^{-nk}.\end{aligned}$$

By L'Hôpital's rule,

$$\limsup_{n\to\infty}\mathbb{P}\big(T(n,k_n)<(1-\epsilon)\beta_\alpha\log n\big)=0. \tag{4.5}$$

In conjunction with (4.2), this completes the proof of the theorem. □

The above theorem describes the achievable trade-off between throughput and delay for perfect fountain coding. The result generalises the case for ARQ; if we take $k_n=1$ for all n, then this corresponds to $\alpha=0$ in the theorem statement. By the theorem

$$\beta_0=\frac{1}{D(0;1-q_{\max})}=\frac{-1}{\log q_{\max}},$$

i.e., ARQ needs $-\log n/\log q_{\max}$ packet transmissions to recover the message. Whilst this is the minimum achievable latency, the throughput achieved vanishes in the limit as the number of receivers, n, tends to infinity. This result is consistent with [62], which shows that the expected number of time-steps required is $\Theta(\log(n))$. The theorem further tells us that throughputs arbitrarily close to capacity are achievable while keeping latencies of the same order, namely logarithmic in n. In particular, if we take $k_n\sim\alpha\log n$, then fountain coding incurs a latency of $\beta_\alpha\log n$ while achieving a throughput of α/β_α. As α increases to infinity, so does β_α, while the ratio α/β_α tends to $1-q\text{max}$, which is the channel capacity. In other words, as throughput approaches capacity, the latency occurred becomes an arbitrarily large multiple of $\log n$.

The relationship between the delay (scaled by $\log n$) and throughput is plotted in Figure 4.2, for three different values of the erasure probability q (and with all $q_i=q$). The figure shows that higher throughputs incur higher delays, and that the delay blows up as throughput approaches capacity. Also, as would be expected, the delay increases with the erasure probability. The graphic is summarising asymptotic results, but can further be interpreted to show that any constant rate can be

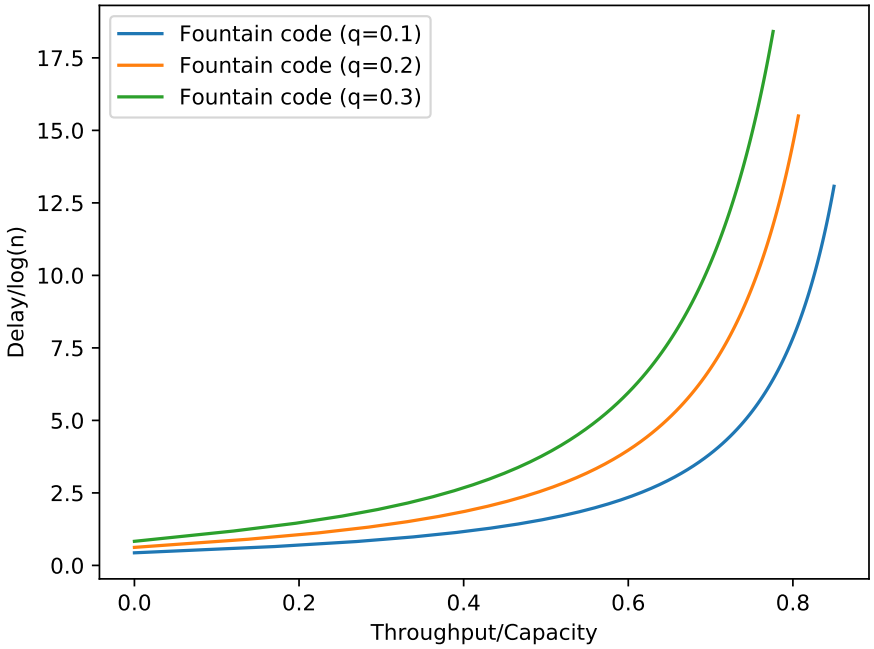


Figure 4.2: Delay vs. throughput of perfect fountain coding.

achieved by fountain coding, with a delay penalty a constant multiple of $\log(n)$, with probability converging to 1 with the number of receivers.

This section concludes with a heuristic calculation of $T(n, k_n)$ based on the Central Limit Theorem (CLT). Even though Theorem 4.1 gives a precise asymptotic expression for $T(n, k_n)$, it is not clear *a priori* how large n and k_n need to be to yield a good approximation. We now present an alternative approximation, for the case in which $q_i = q$, $\forall i \in V$.

Let X_i denote a random variable with the distribution of the number of packet transmissions required for receiver i to receive a single packet. Then, $X_i \sim Geom(1-q)$, and so $\mathbb{E}[X_i] = 1/(1-q)$ and $\text{Var}(X_i) = q/(1-q)^2$. Now $T_i(k)$, defined as the time until receiver i obtains k distinct packets, is the sum of k iid copies of X_i. Hence, for large k, we have by the CLT that

$$\frac{1-q}{\sqrt{kq}}\left(T_i(k) - \frac{k}{1-q}\right) \Rightarrow Z \text{ as } k \to \infty,$$

where $\Rightarrow$ denotes convergence in distribution, and Z denotes a standard normal random variable. Let Φ denote the cdf of Z, i.e., $\Phi(x) = \mathbb{P}(Z \leq x)$.

Now $T(n,k) = \max_{i=1}^{n} T_i(k)$, and the $T_i(k)$ are mutually independent. Hence, $\mathbb{P}(T(n,k) \leq x) = (\mathbb{P}(T_k) \leq x)^n$. Using the CLT to approximate the cdf of $T_i(k)$ as

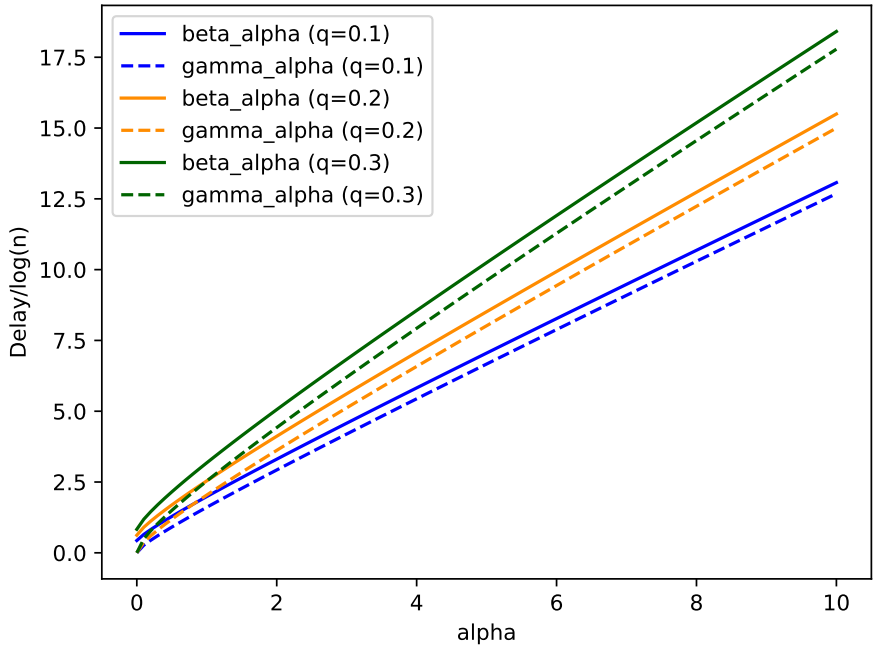


Figure 4.3: Delay coefficients from Theorem 4.1 (β_α) and the CLT approximation in Equation. 4.8 (γ_α).

above, we obtain

$$\mathbb{P}\Big(T(n,k) \le \frac{k+\sqrt{kqx\log n}}{1-q}\Big) \approx \Big(\Phi\left(\sqrt{x\log n}\right)\Big)^n. \tag{4.6}$$

The reason that this is not a limit theorem is that the CLT establishes convergence of distribution in the bulk, whereas we are using it inappropriately to approximate the distribution in the tail.

Next, using the inequality

$$1-\Phi(z) \le \frac{1}{z\sqrt{2\pi}}\exp\Big(-\frac{z^2}{2}\Big),$$

which can be obtained by integrating the normal density by parts, we have from (4.6) that

$$\begin{aligned}&\mathbb{P}\Big(T(n,k) \le \frac{k+\sqrt{kqx\log n}}{1-q}\Big)\\ &\approx \Big(1-\frac{n^{-x/2}}{\sqrt{2\pi x\log n}}\Big)^n \approx \exp\Big(-\frac{n^{1-(x/2)}}{\sqrt{2\pi x\log n}}\Big).\end{aligned}$$

The term in the exponent tends to zero if $x > 2$ and to $-\infty$ if $x < 2$. This suggests the heuristic

$$T(n,k) \approx \frac{k+\sqrt{2kq\log n}}{1-q}. \tag{4.7}$$

Comparing this result with Theorem 4.1, we see that if we fix $\alpha > 0$ and take $k_n = \lceil \alpha \log n \rceil$, then (4.7) suggests $T(n, k_n) \approx \gamma_\alpha \log n$, where

$$T(n, k_n) \approx \gamma_\alpha \log n, \text{ where } \gamma_\alpha = \frac{\alpha + \sqrt{2\alpha q}}{1 - q}. \tag{4.8}$$

This allows direct comparison with Theorem 4.1, which yields the asymptotically correct expression $T(n, k_n) \approx \beta_\alpha \log n$. We have plotted both β_α and γ_α against α in Figure 4.3, which shows that they are very close to each other, but diverge as α increases. The figure leads us to conjecture that $\beta_\alpha \geq \gamma_\alpha$ for all $\alpha \geq 0$. Whilst I could not prove this conjecture, I was able to come very close, and lower bound β_α by something close to γ_α, as shown in the following lemma.

Lemma 4.2. *If $q_i = q \in (0, 1)$, for all $i \in \{1, \ldots, n\}$, then $\beta_\alpha \geq \frac{\alpha + \sqrt{\alpha q}}{1-q}$, $\forall \alpha \in \mathbb{R}^+ \setminus \{0\}$.*

Proof. Let $f(x) = xD(\frac{\alpha}{x}; 1 - q)$, let $\gamma'_\alpha = \frac{\alpha + \sqrt{\alpha q}}{1-q}$. By the definition of β_α, the result follows if $f(\gamma'_\alpha) < 1$. We have

$$\begin{aligned}
f(\gamma'_\alpha) &= \gamma'_\alpha \big(\tfrac{\alpha}{\gamma'_\alpha} \log \big(\tfrac{\alpha}{\gamma'_\alpha (1-q)} \big) \big) + \Big(1 - \tfrac{\alpha}{\gamma'_\alpha} \log \Big(\tfrac{1 - \frac{\alpha}{\gamma'_\alpha}}{q} \Big) \Big) \\
&= \alpha \log \big(\tfrac{\alpha}{\alpha + \sqrt{\alpha q}} \big) + (\gamma'_\alpha - \alpha) \log \big(\tfrac{1}{q} \big(1 - \tfrac{(1-q)\alpha}{\alpha + \sqrt{\alpha q}} \big) \big) \\
&\leq \alpha \big(\tfrac{\alpha}{\alpha + \sqrt{\alpha q}} - 1 \big) + \tfrac{\alpha q + \sqrt{\alpha q}}{1-q} \big(\tfrac{1}{q} \big(1 - \tfrac{(1-q)\alpha}{\alpha + \sqrt{\alpha q}} \big) - 1 \big) \\
&= \alpha \big(\tfrac{\alpha}{\alpha + \sqrt{\alpha q}} - 1 \big) + \tfrac{\alpha q + \sqrt{\alpha q}}{1-q} \tfrac{1-q}{q} \big(1 - \tfrac{\alpha}{\alpha + \sqrt{\alpha q}} \big) \\
&= \alpha \big(\tfrac{\alpha}{\alpha + \sqrt{\alpha q}} - 1 \big) + \alpha \big(1 - \tfrac{\alpha}{\alpha + \sqrt{\alpha q}} \big) + \tfrac{\sqrt{\alpha q}}{q} \big(1 - \tfrac{\alpha}{\alpha + \sqrt{\alpha q}} \big) \\
&= \tfrac{\sqrt{\alpha q}}{q} \tfrac{\sqrt{\alpha q}}{\alpha + \sqrt{\alpha q}} \\
&= \tfrac{\alpha}{\alpha + \sqrt{\alpha q}} \\
&\leq 1.
\end{aligned}$$

□

4.3 Simulation results

Whilst the results of Theorem 4.1 hold asymptotically, and can be expected to give good predictions for large numbers of receivers, it is unclear how large n must be before the results of the previous section are useful. Simulation results are given in this section to complement the rigorous analysis, and show that the bounds are indeed useful for moderate numbers of receivers.

In this section it is assumed that $q_i = q \in [0, 1]$, $\forall i \in \{1, \ldots, n\}$, for the sake of clarity of the desired phenomena in the results. In order to compare the performance

of fountain coding with that of ARQ, broadcasts of messages consisting of varying numbers of messages to varying numbers of receivers over the broadcast erasure channel described in Section 4.1 were simulated using each of the techniques. The simulations were implemented in Python. Each model and parameter combination was simulated for 368640 messages . Simulations were carried out for three different erasure probabilities, $q = 0.1$, 0.2 and 0.3, for $n = 20$ receivers, and for a wide range of message sizes, ranging from $k = 1$ to $k = 100$ packets. In each case, the total number of packet transmissions, $T(n, k)$, required until all receivers were able to decode the message was obtained from the simulations. The excess latency, defined as the amount by which $T(n, k)$ exceeded the expected minimum number of packets, $k/(1-q)$, required to transmit the message to a single receiver, was calculated. This is a measure of the overhead caused by having multiple receivers. We compare how this overhead differs between ARQ and fountain coding, and also how it depends on the message size.

Figure 4.4 compares the average excess latency of fountain coding and ARQ in simulations. Notice how the excess latency of ARQ is much larger than that of fountain coding. Moreover, the excess latency of ARQ grows linearly with the message size k, whereas that of fountain coding grows sub-linearly, approximately as $\sqrt{k}$ as predicted by Equation. (4.8) in Section 4.2.

Figure 4.5 plots the average excess latency observed in the simulations, for erasure probability $q = 0.3$ and $n = 20$ receivers, as a function of the message size k. The theoretical predictions for the same quantity from Theorem 4.1 and the CLT approximation, Equation. 4.8 are also shown on the same plot. Even though the theoretical results are asymptotic, in a limiting regime in which k and n tend to infinity, the figure shows that they give good predictions even for rather small values of k and n.

4.4 Concluding remarks

This chapter considered the problem of broadcasting over erasure channels, and has shown that coding over an increasing number of packets is required for non-vanishing throughput. The baseline solution ARQ has minimal delay, as it is equivalent coding over just a single packet (i.e only one packet remains buffered). However, the number of time-steps required by the algorithm to successfully broadcast a single message to n receivers scales as $\log(n)$. The throughput of the algorithm therefore vanishes with the number of receivers.

In contrast, the number of time-steps perfect fountain coding is able to achieve

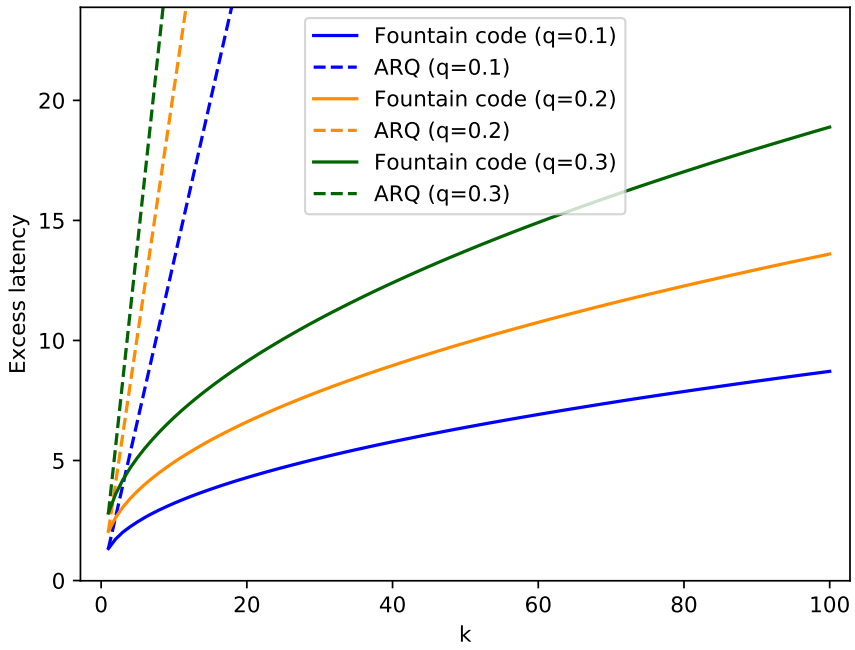


Figure 4.4: Graphic comparing the average excess latency of fountain coding and ARQ in 368,640 simulations, $n = 20$.

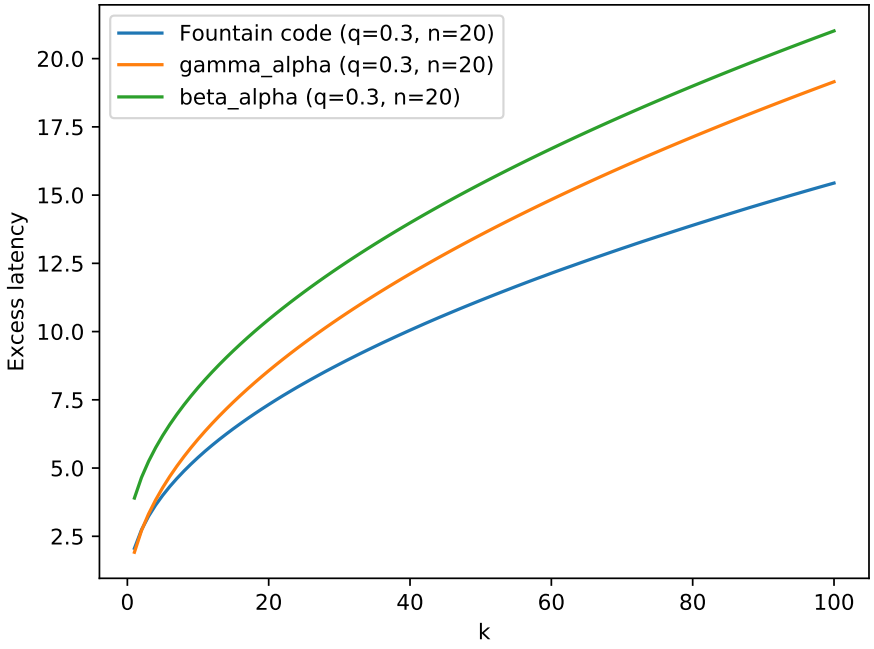


Figure 4.5: Graphic comparing the average excess latency of fountain coding in 368,640 simulations, $n = 20$, with theoretical predictions from Section 4.2.

the successful transmission of $O(\log(n)$ messages also scales as $\log(n)$. Any throughput below capacity is shown to be achievable, but higher throughput results in a larger constant multiplying $\log(n)$. This tradeoff has been fully quantified.

Whilst the absolute minimum latency may be essential for some applications, these systems can only achieve vanishing throughput as the number of receivers increases. To achieve constant throughput, the system must be able to tolerate an amount of delay which increases with the number of receivers.

The main limitation of this work is the assumption of perfect fountain coding; that all k message packets can be obtained by decoding any k coded packets. This in practice is not the case, and indeed several codes (such as Raptor codes [44]) even pay a penalty of overhead in exchange for ease of decoding. It would be useful to extend these results to take such a penalty into account, without loosing their simplicity.

Chapter 5

Allcast over random graphs

The remainder of this thesis is devoted to the allcast problem, in which a collection of n agents wish to mutually exchange messages; all agents have a message or stream of messages to broadcast to every other agent. Whilst focussed on broadcasting or multicasting, the work of Chapter 4 does readily extend to network allcast by simply allocating orthogonal resources (such as time or spectrum) to each agent, and implementing fountain coding for the dissemination of each agents messages. It is clear to see that this approach is throughput optimal for this natural extension of the model. However, this only addresses networks in which pairs of agents have stable, uninterrupted channels. The methods studied rely on the ability of each agent to transmit a message to any other within a short period of time, and a key assumption to ensure this is that the channel erasures are independent across time.

The model introduced in this chapter addresses more practical networks, and aims to better address some of the issues arising in vehicular networks. In particular, the assumption that erasures are independent over time will be relaxed. This relates, for example, to situations where vehicles may be obstructed by obstacles or terrain for lengthy periods. Pairs of agents may have to wait a considerable amount of time for direct communications links to become available. They would have to cooperate with other agents with which viable links did exist, in order to communicate urgent messages during this time, and overcome this slow fading.

This chapter concludes with a baseline solution, with a rigorous asymptotic bound on the number of time-steps required for its completion, which is complemented by Monte-Carlo simulations. Network coding will be introduced as a solution with two implementations given and analysed in Chapters 6 and 7.

5.1 System model

In contrast to Chapter 4, since cooperation between agents to forward messages may be essential, it makes sense to consider connectivity of the entire set of agents rather than just the outgoing links of a single agent. This model considers n agents, which at each time-step t are positioned at nodes on a random digraph $G_t = (V, E_t)$, $|V| = n$, $E_t \subseteq V \times V$. Each edge $(i, j) \in E$ represents an error and delay free communication link of unit capacity between i and j. By choosing edges to be directed, no assumption of channel reciprocity is made.

Communications are synchronised to a common clock; at each time-step $t \in \mathbb{N}$, every node on the graph has the opportunity to transmit a single message to all adjacent nodes. Since this thesis aims to address problems over broadcast media, it is further assumed that when a node on the graph transmits a message, it transmits the same message to every node to which it is adjacent.

We generalise the model presented in [24] which only models static graphs, as follows. The random digraph G_1 is initially realised before communications commence: each edge exists from one node to another i.i.d with probability $p \in (0, 1)$. At each subsequent time-step, each possible edge in $V \times V$ (whether present in the graph or not), is chosen for re-selection i.i.d with probability $\alpha \in [0, 1]$. If an edge is selected for re-selection, then it is re-sampled according to the initial distribution (to be present i.i.d with probability p). It is easy to confirm that the presence of the edges on the graph may equivalently be considered to evolve as independent two-step Markov chains, whose stationary distribution is Bernoulli(p).

At each time-step, the graph is a realisation of the well known Erdős-Rényi graph model. This well studied model is named after the mathematicians who introduced random graph models [82] and had many of the early results in the field [83]. The term commonly refers to two graph models: one in which the number of edges is fixed (as in [82]), and the more commonly studied model in which edges are realised i.i.d with fixed probability (as studied here), which was in fact first introduced by Gilbert [84]. The term Erdős-Rényi will herein be used to refer to the latter of the two models.

Notice that if $\alpha = 1$, then the edges of the graph may be considered to be i.i.d erasure channels, with erasure probability $1 - p$, and we obtain a fast fading model as a special case. Further, if $\alpha = 0$, the graph will remain fixed for all time (which may well model transceiving nodes which are fixed at random positions in dense terrain). Choosing $\alpha \in (0, 1)$ may be considered to model slow fading experienced by transceivers which are moving through terrain, in the sense that the nodes stay

connected or disconnected for extended periods of time (for an average of $\frac{1}{\alpha(1-p)}$ and $\frac{1}{\alpha p}$ time-steps, respectively). When designing algorithms in these chapters, it will be assumed that the length of time nodes have to wait before they can establish a communication link with non-adjacent nodes is intolerable, and that the nodes must in some way *relay* each others messages in order to mitigate this. If $\alpha = 0$ then allcast is *impossible* without relaying, except for the rare case when G_t is complete for all time. If $\alpha = 0$, for convenience and clarity of notation, we drop the subscript t and denote the connectivity graph at every time-step by $G = (V, E)$. For ease and tractability of analysis, all analytical work under this model has been performed under the assumption that $\alpha = 0$, which can be considered the worst case for the model. The more general case will be investigated with Monte Carlo simulations.

This section concludes with two useful results. The first of these concerns the *diameter* of G: the least $d \in \mathbb{N}$ such that for each $i, j \in V$, there exists a path of length d or less from i to j. Although this result is not utilised directly in any proof contained in this thesis (except Theorem 5.2 which follows it), it provides some insight into the model, and is frequently used to justify the construction of the solution algorithms.

Lemma 5.1. *Let d denote the diameter of G. Then $\mathbb{P}(d > 2) \leq \frac{n^2-n}{2}\big(1-p^2\big)^{n-2}$ and $\mathbb{P}(d < 2) \leq p^{\frac{1}{2}n(n-1)}$.*

Remark. By the above lemma, G will have diameter 2 with high probability as $n \longrightarrow \infty$. That is to say, that with high probability, each node will be able to communicate a message to any other with the cooperation of at most one other as a relay.

Proof. For the first claim, let $i, j \in V$, and let $B_{i,j}$ denote the event that there exists no intermediate node $k \in V$ such that $(i, j), (j, k) \in E$. Then

$$\mathbb{P}(B_{i,j}) = \prod_{\substack{k \in V \\ k \neq i,j}} \big(1 - p^2\big) = \big(1 - p^2\big)^{n-2}$$

Clearly $d > 2$ if and only if $B_{i,j}$ is true for at least one pair $i, j \in V$, and there are $\binom{n}{2} = \frac{n^2-n}{2}$ such pairs. Hence, by the union bound

$$\mathbb{P}(d > 2) \leq \frac{n^2 - n}{2}\big(1 - p^2\big)^{n-2}.$$

For the second claim, note that $d < 2$ if and only if the graph is complete, i.e if each of the $\frac{1}{2}n(n-1)$ edges are present, which occurs with probability $p^{\frac{1}{2}n(n-1)}$. □

It will now be shown that at least a constant number of transmission rounds are required, by any method, for successful dissemination of all messages to all agents.

Theorem 5.2. *Suppose each node in G wishes to broadcast a message to all others, and that each message is chosen uniformly at random, independently from all others. Let X denote the earliest transmission round at which every node has done so, by any means (using only transmissions over edges in E). Then*

$$\lim_{n\longrightarrow\infty} \mathbb{P}(X < \tfrac{1-\epsilon}{p}) = 0.$$

Proof. Let A be the event that G has diameter 2. Let X_i be the first round at which node i has received every message (noting that $X = max_{i=1}^{n} X_i$), and let $B_i = \{|N_i^{\text{in}}| < \frac{1}{1-\epsilon}p(n-1)\}$. If A and B_i occur, then after $\frac{1-\epsilon}{p}$ rounds, node i will have received fewer than $n-1$ packets. No method exists which can encode $n-1$ messages to fewer than $n-1$ coded messages, and then uniquely decode the original message, so decoding must be impossible, and so $\mathbb{P}(X_i < \frac{1-\epsilon}{p}|A, B_i) = 0$. We have

$$\begin{aligned}
\mathbb{P}\big(X < \tfrac{1-\epsilon}{p}\big) &= \mathbb{P}\big(X < \tfrac{1-\epsilon}{p}|A\big)\mathbb{P}(A) + \mathbb{P}\big(X < \tfrac{1-\epsilon}{p}|A^{\mathsf{c}}\big)\mathbb{P}(A^{\mathsf{c}}) \\
&\leq \mathbb{P}\big(X < \tfrac{1-\epsilon}{p}|A\big) + p^{\frac{1}{2}n(n-1)} + \frac{n^2-n}{2}\big(1-p^2\big)^{n-2} \\
&\leq \sum_{i=1}^{n} \mathbb{P}\big(X_i < \tfrac{1-\epsilon}{p}|A\big) + p^{\frac{1}{2}n(n-1)} + \frac{n^2-n}{2}\big(1-p^2\big)^{n-2} \\
&\leq \sum_{i=1}^{n} \Big(\mathbb{P}\big(X_i < \tfrac{1-\epsilon}{p}|A, B_i\big)\mathbb{P}(B_i) + \mathbb{P}\big(X_i < \tfrac{1-\epsilon}{p}|A, B_i^{\mathsf{c}}\big)\mathbb{P}(B_i^{\mathsf{c}})\Big) \\
&\qquad + p^{\frac{1}{2}n(n-1)} + \frac{n^2-n}{2}\big(1-p^2\big)^{n-2} \\
&\leq ne^{-(n-1)H(\frac{1}{1-\epsilon}p;p)} + p^{\frac{1}{2}n(n-1)} + \frac{n^2-n}{2}\big(1-p^2\big)^{n-2},
\end{aligned}$$

where the second inequality follows by Lemmas 2.10 and 5.1, and the final inequality follows by Lemma 2.10. Hence

$$\lim_{n\longrightarrow\infty} \mathbb{P}(X < \tfrac{1-\epsilon}{p}) = 0. \qquad \square$$

5.2 Baseline solution: Random message forwarding

The purpose of this and the next two chapters is to show the advantage of Random Linear Network Coding (RLNC) under this model. To that end, this section introduces a *baseline* solution for which coding is not allowed, which will allow comparison of the solutions presented in Chapters 6 and 7 with a reasonable uncoded method. It will be assumed in these chapters that $\alpha = 0$, i.e the graph is static. The baseline solution will be known as *Random message forwarding*, and is defined as follows.

In the first transmission round, as the agents (nodes) have at this stage received no data from their peers, they can do no better than to each broadcast their own message to every other adjacent node; the alternative would for some agents not to use their first transmission round.

Recall from Section 5.1 that nodes must cooperate in order to achieve allcast by relaying each others messages. Without an expensive system of polling neighbours in order to learn the contents of their buffers, there is no way of knowing which packets are required by adjacent nodes, and it is inevitable that some relay transmissions will not be useful. This method is the most obvious solution: each node buffers the messages it receives from adjacent neighbours, and at each time-step broadcasts a randomly selected message from its buffer to its neighbours.

Algorithm 1 Random message forwarding: an uncoded baseline method for allcast on Erdős-Rényi random graphs

In the first time-step, each agent broadcasts its own message.

After each time-step, each agent adds received messages it did not already posses to a buffer.

In each subsequent time-step, each agent selects a message from its buffer at random, and broadcasts it to its neighbours.

Since the applications require rapid dissemination of messages with minimal latency, an asymptotic bound will now be derived on the number of time-steps until all nodes have received all of the messages. Its proof proceeds in two stages, after proving the following lemma

Lemma 5.3. *Let $x \in [0,1)$, let $\lambda \in (0, \frac{1}{x})$. Then*

$$D(\lambda x; x) \geq \lambda x \log(\lambda) + (1-\lambda x)c\big(1-(1-x)^{-c}\big) - \lambda x.$$

Proof. By definition, we have

$$\begin{aligned} D(\lambda x; x) &= \lambda x \log(x) + (1-\lambda x)\log(\tfrac{1-\lambda x}{1-x}) \\ &= \lambda x \log(x) - (1-\lambda x)\big(\log(1-x) + \log\big(\tfrac{1}{1-\lambda x}\big)\big) \\ &\geq \lambda x \log(x) - (1-\lambda x)\big(\log(1-x) + \tfrac{1}{1-\lambda x} - 1\big) \\ &= \lambda x \log(x) - (1-\lambda x)\log(1-x) - \lambda x, \end{aligned}$$

where the inequality follows since $\log(x) \leq x-1$. Now, for $c \in \mathbb{R}^+$, we have

$$\begin{aligned} D(\lambda x; x) &\geq \lambda x \log(\lambda) - (1-\lambda x)c\log\big((1-x)^{-c}\big) - \lambda x \\ &\geq \lambda x \log(\lambda) - (1-\lambda x)c\big(1-(1-x)^{-c}\big) - \lambda x, \end{aligned}$$

again since $\log(x) \leq x-1$, and $\log(1-x) < 0$. □

Whilst the following lemma simply concerns a binomial random variable, it will become clear that this relates to the number of neighbours of a node i which forward i's message after $O(\log(n))$ time-steps. This is conditioned on i having a given number of out-neighbours, and some other assumptions stated in the proof of Theorem 5.5.

Lemma 5.4. *Let* $g : [0,1] \longrightarrow [-1,1]$; $g(\lambda) = 1 - 2\lambda + \lambda\log(\lambda)$. *Let* $p \in (0,1)$, $n \in \mathbb{N}$, *let* $t = \frac{\alpha(1+\epsilon)}{p}\log(n)$, $\alpha > 0$, $\epsilon > 0$, *let* $d = g^{-1}(p)(1-\epsilon)np\Big(1 - \big(1 - \frac{1}{n}\big)^t\Big)$. *Let* $X \sim Bin\Big((1-\epsilon)np, 1 - \big(1 - \frac{1}{n}\big)^t\Big)$. *Then*

$$\mathbb{P}(X < d) < n^{-\alpha(1+\epsilon)}.$$

Remark: It is easy to show that $g(0) = 1$, $g(1) = -1$, and $g'(\lambda) \leq -1$. And since g is also continuous, g is invertible. Although no inverse exists in closed form, g^{-1} may be approximated numerically for practical purposes (its value is of no importance in the following proof).

Proof. For convenience, let $\lambda = g^{-1}(p)$, let $x = 1-(1-\frac{1}{n})^t$. By Lemma 2.10, we have $\mathbb{P}(X < d) \leq \exp(-nD(\lambda x; x))$. It remains to prove that $D(\lambda x; x) \geq \alpha(1+\epsilon)\log(n)$, and we will make use of Lemma 5.3 to do this.

Since $(1 - \frac{1}{n})^t \geq 1 - \frac{t}{n}$, and since $\log(\lambda) < 0$, we have $n\lambda x\log(\lambda) \geq \lambda t\log(\lambda)$, and

$$\begin{aligned} n(1-\lambda x)t\big(1 - (1-x)^{-t}\big) &= n(1-\lambda x)t\big(1 - (1 - \tfrac{1}{n})\big) \\ &= (1-\lambda x)t. \end{aligned}$$

Finally, again since $(1 - \frac{1}{n})^t \geq 1 - \frac{t}{n}$, we have $-n\lambda x \geq -\lambda t$. Hence, by Lemma 5.3, we have

$$\begin{aligned} nD(\lambda x; x) &\geq \lambda t\log(\lambda) + (1-\lambda x)t - \lambda t \\ &= t(\lambda\log(\lambda) + 1 - \lambda x - \lambda). \end{aligned}$$

Substituting for t and x gives

$$\begin{aligned} nD(\lambda x; x) &\geq \tfrac{\alpha(1+\epsilon)}{p}\big(\lambda\log(\lambda) + 1 - \lambda + \lambda(1 - \tfrac{1}{n})^t - \lambda\big)\log(n) \\ &\geq \tfrac{\alpha(1+\epsilon)}{p}\big(1 - 2\lambda + \lambda\log(\lambda)\big)\log(n) \\ &= \tfrac{\alpha(1+\epsilon)}{p}g\big(g^{-1}(p)\big)\log(n) \\ &= \tfrac{\alpha(1+\epsilon)}{p}\log(n). \end{aligned}$$

□

As well as making previous arguments precise, the following theorem shows that a further $O(\log(n))$ time-steps are sufficient for every agent to receive every message with high probability.

Theorem 5.5. *Consider the model from Section 5.1. Suppose the nodes employ Algorithm 1. Let X denote the number of time-steps before every node has received every message. Let $\epsilon \in \mathbb{R}^+$. Then*

$$\lim_{n\longrightarrow\infty} \mathbb{P}\big(X > \tfrac{3(1+\epsilon)}{p}\log(n)\big) = 0.$$

Remark. Since G has diameter 2 with high probability (by Lemma 5.1), and each node may communicate with any other over at most two hops, it would suffice for each node to select messages from a buffer of the messages received in the first round only (these are the only messages required by their out-neighbours). However, we do not focus on this method as it is less robust, and cannot be applied more generally to graphs with larger diameter.

Proof. Fix $i, j \in V$. In order to bound the number of time-steps before j receives i's packet, we begin by bounding the number of nodes Y which have forwarded i's message after $\frac{1+\epsilon}{p}\log(n)$ time-steps. Let $d^* = \frac{\log\left(\frac{\sqrt{1+\epsilon}-1}{2\sqrt{1+\epsilon}}\right)}{\log(1-p)}$. Notice that Y is distributed exactly as X is in Lemma 5.4, and that $d \longrightarrow \infty$ as defined in the lemma, whilst here d^* is constant). Hence, as a consequence of the lemma, there exists $N \in \mathbb{N}$ such that $(n > N) \implies \big(\mathbb{P}(Y > d^*) < n^{-(1+\delta)}\big)$. We will henceforth assume that $n > N$, as this is sufficient to prove the claim, and assume that there are exactly d^* such neighbours of i.

It remains to prove that a further $\frac{2(1+\epsilon)}{p}\log(n)$ time-steps are sufficient for j to receive i's message. By Lemma 2.10, after the first $\frac{1+\epsilon}{p}\log(n)$ time-steps, at least $\frac{1}{\sqrt{1+\epsilon}}np(1-(1-p)^{d^*}) = np\frac{\frac{1}{2}\sqrt{1+\epsilon}+\frac{1}{2}}{\sqrt{1+\epsilon}}$ in-neighbours of j will have received i's message directly from a neighbour of i, with probability at least $1 - \exp\Big(-n\frac{\frac{1}{2}\sqrt{1+\epsilon}+\frac{1}{2}}{\sqrt{1+\epsilon}}D(\frac{p}{\sqrt{1+\epsilon}};p)\Big)$. We now assume there are exactly this many, neglect other sources of i's message for these nodes, and assume that they are the only sources for j of i's message. The probability of a further $2\frac{1+\epsilon}{p}\log(n)$ time-steps being insufficient is therefore at most

$$\begin{aligned}\left(\big(1-\tfrac{1}{n}\big)^{\frac{1}{\sqrt{1+\epsilon}}np(1-(1-p)^{d^*})}\right)^{2\frac{1+\epsilon}{p}\log(n)} &\leq \exp\Big(-\tfrac{1}{n}\cdot np\frac{\frac{1}{2}\sqrt{1+\epsilon}+\frac{1}{2}}{\sqrt{1+\epsilon}}\cdot 2\tfrac{1+\epsilon}{p}\log(n)\Big)\\ &\leq n^{-(1+\epsilon+\sqrt{1+\epsilon})},\end{aligned}$$

using the inequality $1 + x \leq e^x$. Using the union bound over j then i, and using the law of total probability, we obtain

$$\mathbb{P}\big(X > \tfrac{3(1+\epsilon)}{p}\log(n)\big) \leq n(n-1)n^{-(1+\sqrt{1+\epsilon})} + n\cdot n^{-(1+\epsilon)}$$

$$+ n(n-1)\exp\Big(-n\frac{\frac{1}{2}\sqrt{1+\epsilon}+\frac{1}{2}}{\sqrt{1+\epsilon}}D(\frac{p}{\sqrt{1+\epsilon}};p)\Big).$$

The proof follows by taking limits as $n \longrightarrow \infty$. □

5.3 Simulation results

Whilst the bounds given in Section 5.2 hold asymptotically, these results give no guarantee over the performance of the algorithm for small networks. Monte Carlo simulations are presented in this section to complement the rigorous analysis, showing that the bounds converge quickly and are useful for small n. The graphics in this section give a comparison of 10,000 simulations with the bound given in Theorem 5.5. Caps on the whiskers of the box plots have not been plotted, because the true minima and maxima are 1 and ∞, respectively, and will not be attained in simulations. The whiskers do however illustrate the full range of simulation results.

Figure 5.1 has a logarithmic x-axis scale, so that the asymptotic upper bound may be plotted as a straight line. Notice that the bound and medians do appear to diverge with n, which does suggest that there may be some slack in the upper bound. However, the medians do seem to lie along a straight line, suggesting that the number of timesteps required by algorithm 1 does scale as $\log(n)$ as expected. The simulations suggest the slack is in the constant multiplying $\log(n)$. Notice also that the whiskers do exceed the bound. This does not contradict Theorem 5.5 as the bound only holds asymptotically. The amount by which the whiskers exceed the bound does seem to decrease with n. Recall from Section 5.1 that every agent broadcasts a message in each timestep, and so the overall number of transmissions is increasing as $n\log(n)$.

Figure 5.2 has been plotted with a reciprocal x-axis scales, again so that the bound can be plotted as a straight line. For $p = 1$, every realisation of the graph is the complete graph, for which a single timestep is clearly sufficient. Hence, the line should pass through $(1, 1)$. Clearly this is not the case, but this does explain why the medians appear to grow more clearly nearer $p = 1$. For smaller p, there does not seem to be much divergence, suggesting the scaling in p is correct. The simulations broadly match the theory for all p, despite the theory holding asymptotically.

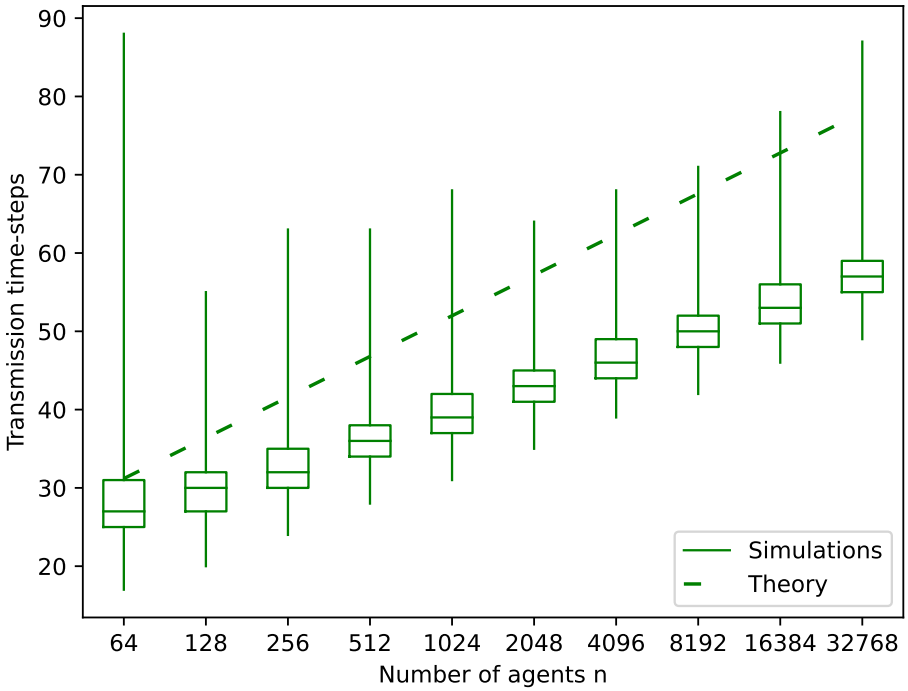


Figure 5.1: Graphic summarising 10,000 Monte Carlo simulations of Algorithm 5.5, in which agents achieve network allcast by broadcasting messages randomly selected from a buffer of those received previously. The network consists of n agents, with edge connection probability $p = 0.4$. The n-axis scale is logarithmic. The asymptotic bound given in Theorem 5.5 is plotted with broken line, whilst the simulations are plotted as box plots. The y-axis shows the number of transmission rounds, in which each agent broadcasts a message to its neighbours, taken before every agent receives every message. The graphic shows that the simulations broadly obey the asymptotic upper bound. The growing discrepancy between the medians and the bound, and the seemingly linear trend of the medians, suggests that there may be some slack in the constant multiplying $\log(n)$ in the bound. Notice that the x-axis shows the number of *time-steps*, in which each agent broadcasts a message. The total number of transmissions is n times that shown on this axis.

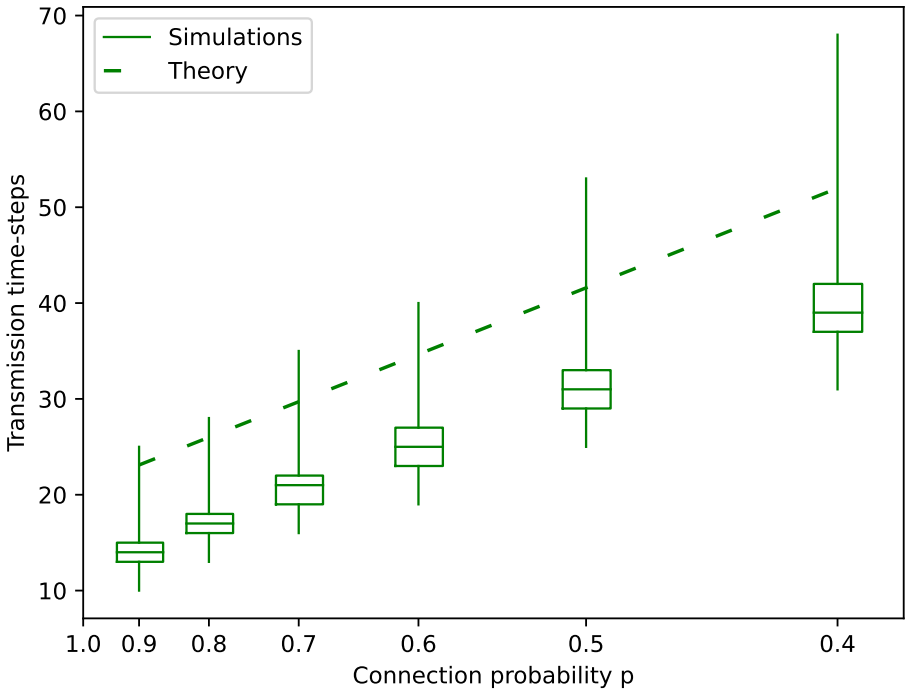


Figure 5.2: Graphic summarising 10,000 Monte Carlo simulations of Algorithm 5.5, in which agents achieve network allcast by broadcasting messages randomly selected from a buffer of those received previously. The network consists of $n = 1024$ agents, with varying edge connection probabilities p. The p axis is plotted on a reciprocal scale. The asymptotic bound given in Theorem 5.5 is plotted with broken line, whilst the simulations are plotted as box plots. The y-axis shows the number of transmission rounds, in which each agent broadcasts a message to its neighbours, taken before every agent receives every message. The simulations broadly obey the bound for all p. Other than a steepening of the simulation graph towards $p = 1$, the scaling does appear to be $\frac{1}{p}$.

Chapter 6

Network coding over random graphs: a chunking method

The main issue with the random forwarding method of Chapter 5 is that each packet selected for transmission by a particular node may not be useful to all the nodes adjacent to it (if any), whilst the transmitting node may possess packets at that time-step which its less fortunate neighbours may not. This chapter will introduce a method based on Random Linear Network Coding (RLNC), in which coded packets are formed by taking random linear combinations of buffered packets received from other agents. In this way, each node may communicate information about multiple packets in each time-step.

The design of the code will take advantage of the fact that, as a consequence of Lemma 5.1, the graph G will have diameter 2 with high probability as $n \longrightarrow \infty$. That is to say that any node can communicate its message to any other over a path of at most two hops. It will suffice, therefore, for each node to forward the messages of its in-neighbours to all of its out-neighbours.

In the first time-step, as in the random message forwarding method of Chapter 5, each agent transmits its own message (as it can do no better). Each agent also buffers these first-round messages from its in-neighbours, keeping them separate from all subsequently received messages. These, and only these messages, will be used by the agent to form coded packets for future transmission. Coding over messages which each agent receives in subsequent time-steps is possible, and could allow generalisation of the method to graphs of diameter greater than 2, but the method does not for ease of analysis. Notice that no delay will be incurred in buffering as was the case in Chapter 4, as the buffer will be filled in one round, whilst good use of the channel was made by all.

6.1 Sparse random matrices over finite fields

Over time, each agent will accumulate a set of random linear combinations of the messages from the other agents. Once their set is full rank, they may decode the messages by solving the resulting linear system. To determine decodability for a particular agent, we may consider the rank of the matrix whose rows are formed from the coefficients of the linear combinations it received (including the first-round messages). The computational expense of encoding and decoding is a barrier to application of RLNC in general, so it will be of interest to minimise the average number of messages included in each linear combination.

For convenience, we now introduce two definitions. The ranks of random matrices are studied in [25]. The article considers $n \times n$ matrices with entries chosen i.i.d from $\mathbb{F}_q$. Each entry is chosen to be zero with fixed probability π, and uniformly from the field otherwise. The authors study the *defect* d of the matrix, which they define to be the difference between n and the rank of the matrix. The article shows that $\mathbb{E}(q^d) = O(1)$, so long as the probability of each entry being non-zero is at least $\frac{\log(n)}{n}$. Therefore, by Jensen's inequality, the expected defect is upper-bounded by a constant. Moreover, by Markov's Inequality, $\mathbb{P}(d > k) \leq \frac{A}{q^k}$.

The RLNC solution suggested by the results of [25] is for agents to broadcast *sparse* random linear combinations of their neighbours' messages (obtained in the first round). After $\lceil \frac{1}{p} \rceil$ time-steps, the coefficient matrices at each agent would have at least as many rows as columns with high probability. If the coefficients are chosen from $\mathbb{F}_2$, clearly by [25, Corollary 2.4], the probability of the defect exceeding some multiple of $\log(n)$ will be upper-bounded by $o(n^{-1})$. By the union bound, each agent would then have at least $n - \log(n)$ linearly independent linear combinations with high probability, and a subset of the messages received by each agent in a small number of additional time-steps should increase the rank of each matrix to n. There are two problems with this idea, however.

Each agent may only code over messages received from its immediate neighbours in the first round. This means that the expected number of non-zero entries per row will now be $np\pi$ rather than $n\pi$. The obvious solution is to increase π by a factor of $\frac{1}{p}$, but unfortunately this does not mitigate the other issue incurred by the graph. If the edge connection probability $p < 0.5$, then each agent will receive more than one coded message from each of its neighbours. Each column in the matrix corresponds to the message of a particular agent, and a non-zero entry in a row and column indicates that the corresponding neighbour included the corresponding agent's message in that particular linear combination. Note that if

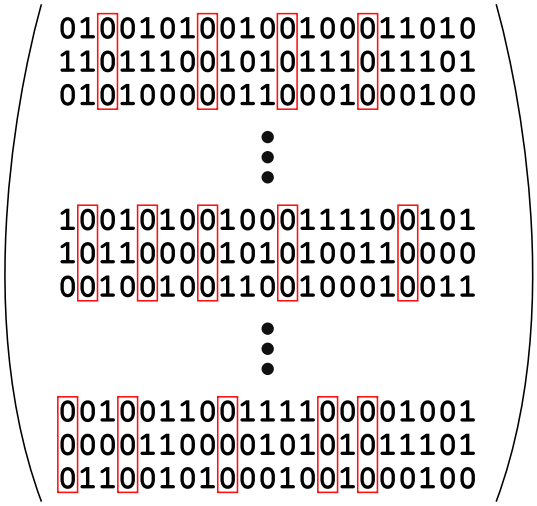


Figure 6.1: Graphic demonstrating the correlation between entries of each agent's received coefficient matrix, which is induced by the random graph. Notice the red boxes, showing sections of the matrix which are forced equal to zero, and whose entries are not independent.

a particular neighbour j does not possess the message of a given agent k, then all entries of column k will be zero in rows which correspond to agent i. This breaks the assumption of [25] that the entries of the matrix must be i.i.d, meaning that this result cannot be used directly to prove any results for this method. This issue is illustrated in Figure 6.1. In the next section, a novel RLNC method is introduced which eliminates the correlation between rows of the matrix, allowing application of these results to prove results for this new method. Chapter 7 overcomes this by presenting new results on the ranks of these random matrices, with analysis inspired by [25].

6.2 A chunking method

To ensure that there is no correlation between rows of the matrix, a new method is introduced. Ahead of transmission time, the set of nodes is partitioned into $\sigma \approx \frac{1}{p} - 1$ subsets S_j. In the first time-step, the agents broadcast their own message as before, In the next time-steps $1 < t \leq \sigma + 1$, the agents broadcast random linear combinations of messages from agents in the intersection of subset S_{t-1} and their in-neighbourhood. Coding is performed over $\mathbb{F}_2$, as the choice of field does not affect the analysis, and the decoding complexity and overhead due to sharing coefficients increases with the order of the field [50]. The coefficients of these linear combinations is taken to be 1 i.i.d with probability π_2, and zero otherwise. These will be termed as *partial* linear combinations. This leads to a block-diagonal structure in

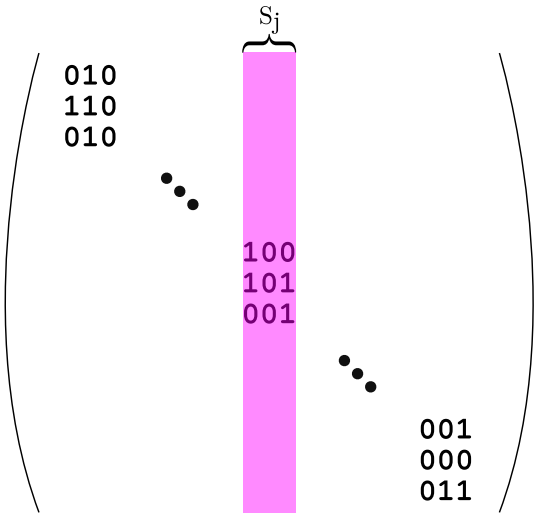


Figure 6.2: Graphic demonstrating the block diagonal structure of the decoding matrices, which is a direct result of coding over one subset of the agents at a time. Entries of the submatrices on the diagonal are i.i.d. Entries in column i represent the inclusion of the corresponding agent's message in the linear combination its row corresponds to. A subset of the columns are highlighted and labelled, to illustrate the subsets S_j of the agents.

the corresponding section of each coefficient matrix, which is illustrated in Figure 6.2. The sub-matrices of the diagonals of these will each contain a single row from each in-neighbour, and hence their entries are i.i.d, and equal 1 with probability $\pi_2 p$. The results of [25] are now applicable to the sub-matrices. It will be shown that the defect of the these overall matrices consisting of these messages, and those from the first round, will not exceed some multiple of $\log(n)$ with high probability. The name given to the method in this chapter owes to its slight similarity to other chunking methods (which is performed for very different reasons, see Chapter 3 for details), in the sense that the set of messages for communication are "chunked".

In one additional, final time-step, a random linear combination of messages from all neighbours will be broadcast by each agent, with coefficients chosen to be 1 with probability π_1 and zero otherwise. This will be termed as a *full* linear combination. A potentially unlimited number of these may be broadcast, as in Section 6.3. However, the following analysis will show that only one will be necessary for each agent to receive enough additional full linear combinations to form a linearly independent set. There must be a limit to the number of transmissions in practice, otherwise the algorithm will never terminate. To enable decoding, the coefficients of all linear combinations are included as packet headers. This assumes that a unique labelling of each node can be arranged ahead of time.

To summarise, the agents employ Algorithm 2, as follows:

Algorithm 2 Chunking RLNC algorithm for allcast on Erdős-Rényi random graphs

1: In the first time-step, each agent broadcasts its own message. It also stores each message received in the first round in a buffer. Let N_v^{in} denote the set of packets received by node v in the first round.

2: In time-steps $1 < t\sigma + 1$, each agent broadcasts a random linear combination, over $\mathbb{F}_2$ of the messages in its buffer from nodes in S_{t-1}, where the sets S_t are a partition of the vertex set V negotiated ahead of time. Let $X_{vw}(t)$ denote the coefficient assigned by node v to node w's packet in round t. Then, $X_{vw}(t)$ are mutually independent Bernoulli random variables, and

$$\mathbb{P}(X_{vw}(t) = 1) = \begin{cases} \pi_2, & w \in \mathrm{N}_v^{\text{in}} \cap S_t, \\ 0, & \text{otherwise.} \end{cases}$$

3: In the final time-step $t = \sigma + 2$, each agent broadcasts a random linear combination, over $\mathbb{F}_2$, of packets it received in the first round, computed as follows. Let $X_{vw}(t)$ denote the coefficient assigned by node v to node w's packet in round t. Then, $X_{vw}(t)$ are mutually independent Bernoulli random variables, and

$$\mathbb{P}(X_{vw}(t) = 1) = \begin{cases} \pi_1 & w \in \mathrm{N}_v^{\text{in}}, \\ 0, & \text{otherwise.} \end{cases}$$

An upper-bound on the number of time-steps required will now be presented.

Theorem 6.1. *Suppose the modelling assumptions of Section 5.1 hold. Let $\epsilon \in \mathbb{R}^+$, and suppose the agents employ Algorithm 2, with the following choices of parameters: $\pi_1 = \frac{\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil \left((1+\epsilon)\log_2(n)+1\right)}{(1-\epsilon)np^2} = O\big(\frac{\log(n)}{n}\big)$, $\pi_2 = \frac{\log\left(|S_j|-(\lceil(1-\epsilon)p(|S_j|-1)\rceil+1)\right)}{\left(|S_j|-(\lceil(1-\epsilon)^2p(|S_j|-1)\rceil+1)\right)p} = O\Big(\frac{\log(n)}{n}\Big)$. A total of $\sigma = \left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil$ partial linear combinations will be broadcast by each agent.*

Let the set of agents V be enumerated by natural numbers, such that $V = 1, \ldots, n$. Before transmissions commence, the agents are grouped into $\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil$ sets, as follows:

$$S_j = \left\{ i \in V : (j-1)\left\lceil \frac{n}{\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil} \right\rceil < i \leq \min\left\{ j \cdot \left\lceil \frac{n}{\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil} \right\rceil, n \right\} \right\},$$

and this partition is known by all nodes.

Let X denote the event that all nodes may decode every message after $\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil + 2 \approx \left\lceil \frac{1}{p} \right\rceil + 1$ time-steps. Then

$$\lim_{n\longrightarrow\infty} \mathbb{P}(X^{\mathsf{c}}) = 0.$$

Remark: The reader may verify that the Theorem also holds for larger probabilities π_1', π_2', so long as $\pi_1 \leq \pi_1' \leq 0.5$ and $\pi_2 \leq \pi_2' \leq 0.5$. The case for π_1 is trivial, and the case for π_2 follows since the expected rank of each matrix $\check{M}_i^j$ in the following proof is monotone increasing for probabilities p_i' in this interval [25], meaning Corollary 2.4 of [25] also holds.

Proof. Fix $i \in V$, let X_i be the event that i may decode all messages after the final transmission round, and let $j^* \in V$ such that $i \in S_{j^*}$. For $j \neq j^*$, let $A_i^j = \big\{|\{k \in S_j : (k,i) \in E\}| \geq (1-\epsilon)p|S_j|\big\}$, and let $A_i^{j^*} = \big\{|\{k \in S_{j^*} : (k,i) \in E\}| \geq (1-\epsilon)p(|S_{j^*}|-1)\big\}$. By Lemma 2.10 and the union bound,

$$\mathbb{P}\big(\cup_j A_i^{j\mathsf{c}}\big) \leq \alpha n e^{-\beta n},$$

where $\alpha, \beta \in \mathbb{R}^+$.

Note that the packets sent in the first round, as well as i's own packet, may be considered to be (trivial) linear combinations. Let M_i be a matrix whose rows are the coefficients of these linear combinations, and those received in the partial rounds, with rows from each round and the trivial linear combinations whose node is a member of the corresponding set S_j grouped together in blocks. The matrix M_i will have n columns, and $\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil$ blocks, each of which will be formed by a single matrix of $|S_j|$ columns, which we will label M_i^j, and zeros in the remaining columns. Let δ_i^j be the defect of M_i^j, and δ_i be the defect of M_i, where we define the defect of a matrix to be the difference between its rank and the number of columns (as in [25]). Since the columns of the M_i^j do not overlap, we have $\delta_i = \sum_j \delta_i^j$.

If $\cap_j A_i^j$ occurs, then $\big|N_i^{\text{in}}\big| \geq \lceil (1-\epsilon)(n-1)p \rceil$. We will assume that i discards any packets from its neighbours from each S_j in excess of $\lceil (1-\epsilon)p|S_j| \rceil$, or $\lceil (1-\epsilon)p\big(|S_{j^*}|-1\big) \rceil$ in the case of j^*, and note that i must receive at least this many packets in either case. We also assume that i discards sparse packets if there are more of these and trivial packets in total than $|S_j|$, so that each M_i is square (padding M_i^j with all zero rows as necessary). For $j \notin \big\{j^*, \lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \rceil\big\}$, we have

$$\begin{aligned}
\big(1-(1-\epsilon)p\big)\left\lceil \frac{n}{\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil} \right\rceil &\leq \big(1-(1-\epsilon)p\big)\left(\frac{(1-\epsilon)np}{1-(1-\epsilon)p} + 1\right) \\
&= 1-(1-\epsilon)p + (1-\epsilon)np \\
&= 1 + (1-\epsilon)(n-1)p.
\end{aligned}$$

Hence

$$(n-1)(1-\epsilon)p + (1-\epsilon)p\left\lceil \frac{n}{\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil} \right\rceil \geq \left\lceil \frac{n}{\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil} \right\rceil - 1,$$

and

$$\lceil (n-1)(1-\epsilon)p \rceil + \left\lceil (1-\epsilon)p\left\lceil \frac{n}{\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil} \right\rceil \right\rceil \geq \left\lceil \frac{n}{\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil} \right\rceil - 1. \tag{6.1}$$

Hence, at most one all zero row of padding is added to each of these M_i^j. By a similar argument, in the case that $j^* \neq \left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil$,

$$1 + \lceil (n-1)(1-\epsilon)p \rceil + \left\lceil (1-\epsilon)p\left(\left\lceil \frac{n}{\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil} \right\rceil - 1 \right) \right\rceil \geq \left\lceil \frac{n}{\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil} \right\rceil - 1. \tag{6.2}$$

So again, at most one zero row of padding is added to $M_i^{j^*}$. In the case of the final set, it can easily be shown using Equations 6.1 and 6.2 that at most one all zero row will be added to $M_{\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil}$, whether the set contains i or not.

We now wish to calculate the δ_i and δ_i^j. We may rearrange columns and rows of M_i^j so that the first $\lceil (1-\epsilon)p|S_j| \rceil$ (or $\lceil (1-\epsilon)p\big(|S_{j^*}|-1\big) \rceil$) columns contain an identity matrix, followed by all-zero columns. The remaining rows in these first columns may then be eliminated. We assume that the padding row (if added) is immediately after the identity rows, we label the square matrix contained in the rightmost columns of the remaining rows $\check{M}_i^j$, and the defect of this matrix $\check{\delta}_i^j$. Notice that the columns of $\check{M}_i^j$ are independent, as the coefficients are chosen independently. The rows of $\check{M}_i^j$ are also independent, as the memberships of edges in E are independent events. The probability of an element of $\check{M}_i^j$ equalling 1 is then

$$\frac{\log\big(|S_j| - (\lceil (1-\epsilon)p(|S_j|-1) \rceil + 1)\big)}{\big(|S_j| - (\lceil (1-\epsilon)p(|S_j|-1) \rceil + 1)\big)p} \cdot p = \frac{\log\big(|S_j| - (\lceil (1-\epsilon)p(|S_j|-1) \rceil + 1)\big)}{\big(|S_j| - (\lceil (1-\epsilon)p(|S_j|-1) \rceil + 1)\big)}.$$

Since this is above the minimum probability required for their result, we may use Corollary 2.4 from [25] to deduce that

$$\mathbb{P}(\delta_i^j \geq (1+\epsilon)\log_2(n) + 1 | \cap_j A_i^j) \leq \mathbb{P}(\check{\delta}_i^j \geq (1+\epsilon)\log_2(n) | \cap_j A_i^j) \leq \lambda_i^j n^{-(1+\epsilon)},$$

for some $\lambda_i^j \in \mathbb{R}^+$. Let $B_i = \left\{ \delta_i \leq \left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil \Big((1+\epsilon)\log_2(n) + 1 \Big) \right\}$. By the union bound, we have $\mathbb{P}(B_i^{\mathsf{c}} | \cap_j A_i^j) \leq \lambda_i n^{-(1+\epsilon)}$, where $\lambda_i = \sum_j \lambda_i^j \in \mathbb{R}^+$.

Node i may decode all messages if and only if the combination of the rows of M_i and the dense packets received by i is full rank. Assuming B_i occurs, we may reduce M_i to a matrix $\check{M}_i$, containing $n - \left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil \Big((1+\epsilon)\log_2(n) + 1 \Big)$ linearly independent

rows, with an identity matrix in the first $n - \left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil \Big((1+\epsilon)\log_2(n)+1 \Big)$ rows and columns, and with all zero rows afterwards. We may then search the set of dense packets for ones which are linearly independent with the non-zero rows already in $\check{M}_i$, discarding those that are not, and adding those that are to $\check{M}_i$, re-arranging the matrix in the same way each time. If we are able to add $\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil \Big((1+\epsilon)\log_2(n)+1 \Big)$ rows to $\check{M}_i^j$ in this way, then the system is full rank.

Let $C_i^j = \big\{ |\{k \in N_i^{\text{in}} : (j,k) \in E\}| \geq (1-\epsilon)np^2 \big\}$, let J_i be the set of nodes corresponding to non-identity columns in $\check{M}_i$. Then $\mathbb{P}(\bigcup_{j \in J_i} C_i^{j\mathsf{c}} | B_i) \leq \left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil \Big((1+\epsilon)\log_2(n)+1 \Big) e^{-\gamma n}$.

Following the proof of Theorem 6.3 in [25], we notice that a dense packet is linearly dependent with the existing rows of $\check{M}_i$ if and only if it is a member of the subspace spanning its rows. If we choose the first $n - \delta_i$ columns arbitrarily, the final δ_i columns are uniquely determined by them; differing in any one of the final δ_i columns then implies linear independence. Notice that an element of a dense row may only be equal to 1 if the node which sent it is adjacent to the corresponding node. For each of the determined columns, there are at least $(1-\epsilon)np^2$ such nodes, but these in general will not be distinct for each column. If we limit our search to $(1-\epsilon)np^2$ packets, we can guarantee to find a packet which is adjacent to any column on each draw.

The probability that a given packet in this set differs in one particular column is at least π_1, and these probabilities are independent amongst packets and columns, as coefficients are chosen independently amongst nodes. We may now view successfully adding $\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil \Big((1+\epsilon)\log_2(n)+1 \Big)$ rows to $\check{M}_i$ in this way as achieving $\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil \Big((1+\epsilon)\log_2(n)+1 \Big)$ successes in $(1-\epsilon)np^2$ trials, i.e

$$\begin{aligned}
\mathbb{P}(X_i^{\mathsf{c}} | \cap_j A_i^j, B_i, \cap_{j \in J} C_i^j) &\leq \mathbb{P}\left(\text{Bin}\left((1-\epsilon)np^2, \frac{\left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil \big((1+\epsilon)\log_2(n)+1 \big)}{(1-\epsilon)^2 np^2} \right) \right. \\
&\qquad \left. < \left\lceil \frac{1-(1-\epsilon)p}{(1-\epsilon)p} \right\rceil \Big((1+\epsilon)\log_2(n)+1 \Big) \right) \\
&\leq e^{-an},
\end{aligned}$$

where $a \in \mathbb{R}$. Now

$$\begin{aligned}
\mathbb{P}(X^{\mathsf{c}}) &\leq \sum_{i=1}^{n} \mathbb{P}(X_i^{\mathsf{c}}) \\
&= \sum_{i=1}^{n} \big(\mathbb{P}(X_i^{\mathsf{c}} | \cap_j A_i^j) \mathbb{P}(\cap_j A_i^j) + \mathbb{P}(X_i^{\mathsf{c}} | \cup_j A_i^{j\mathsf{c}}) \mathbb{P}(\cup_j A_i^{j\mathsf{c}}) \big) \\
&\leq \sum_{i=1}^{n} \mathbb{P}(X_i^{\mathsf{c}} | \cap_j A_i^j) + \alpha n e^{-\beta n}
\end{aligned}$$

$$= \sum_{i=1}^{n} \left(\mathbb{P}(X_i^{\mathsf{c}} | \cap_j A_i^j, B_i) \mathbb{P}(B_i | \cap_j A_i^j) + \mathbb{P}(X_i^{\mathsf{c}} | \cap_j A_i^j, B_i^{\mathsf{c}}) \mathbb{P}(B_i^{\mathsf{c}} | \cap_j A_i^j) \right) + \alpha n e^{-\beta n}$$
$$= \sum_{i=1}^{n} \mathbb{P}(X_i^{\mathsf{c}} | \cap_j A_i^j, B_i) + \alpha n e^{-\beta n} + \lambda n^{-\epsilon},$$

where $\lambda \in \mathbb{R}$. We have

$$\begin{aligned}
\mathbb{P}(X^{\mathsf{c}}) &\leq \sum_{i=1}^{n} \big(\mathbb{P}(X_i^{\mathsf{c}} | \cap_j A_i^j, B_i, \cap_{j \in J_i} C_i^j) \mathbb{P}(\cap_{j \in J_i} C_i^j | \cap_j A_i^j, B_i) \\
&\quad + \mathbb{P}(X_i^{\mathsf{c}} | \cap_j A_i^j, B_i, \cup_{j \in J_i} C_i^{j\mathsf{c}}) \mathbb{P}(\cup_{j \in J_i} C_i^{j\mathsf{c}} | \cap_j A_i^j, B_i) \big) + \alpha n e^{-\beta n} + \lambda n^{-\epsilon} \\
&\leq \sum_{i=1}^{n} \mathbb{P}(X_i^{\mathsf{c}} | \cap_j A_i^j, B_i, \cap_{j \in J_i} C_i^j) + \alpha n e^{-\beta n} + \lambda n^{-\epsilon} + b(n(1+\epsilon)\log_2(n) + n) e^{-\gamma n} \\
&\leq \alpha n e^{-\beta n} + \lambda n^{-\epsilon} + b(n(1+\epsilon)\log_2(n) + n) e^{-\gamma n} + cne^{-dn},
\end{aligned}$$

where $b, c, d \in \mathbb{R}^+$. Hence

$$\lim_{n \longrightarrow \infty} \mathbb{P}(X^{\mathsf{c}}) = 0.$$

□

6.3 Simulation results

Whilst the bounds given in this section hold asymptotically, these results give no guarantee over the performance of the algorithm for small networks. Monte-Carlo simulations are presented in this section to complement the rigorous analysis. The figures in this section give a comparison of 10,000 simulations with the bound given in Theorem 6.1. Caps on the whiskers of the box plots have not been plotted, because the true minima and maxima are 1 and ∞, respectively, and will not be attained in simulations. The whiskers do however illustrate the full range of simulation results. Also note the box-plots have frequently collapsed to lines with whiskers, because the inter-quartile range is zero in these cases. The y-axis in all figures shows the number of timesteps (in which each agent broadcasts a single message) after which all agents can decode all messages.

Figure 6.3 shows the performance against the edge connection probability p, plotted with a logarithmic scale on the x-axis. The simulations match theory exceptionally well. The graph does highlight a transition at $p = 0.5$. Figure 6.4 shows simulation results for additional values of p in this region. The sharp transition is not surprising, it is a direct consequence of the design of the algorithm. For $p > 0.5$, the agents are partitioned into only $\sigma = 1$ subset, whereas for $\frac{1}{3} < p < \frac{1}{2}$, the agents are partitioned into $\sigma = 2$ subsets. Whilst decoding may be possible without the transmission of a full linear combination, this event is not exhibited in 10,000

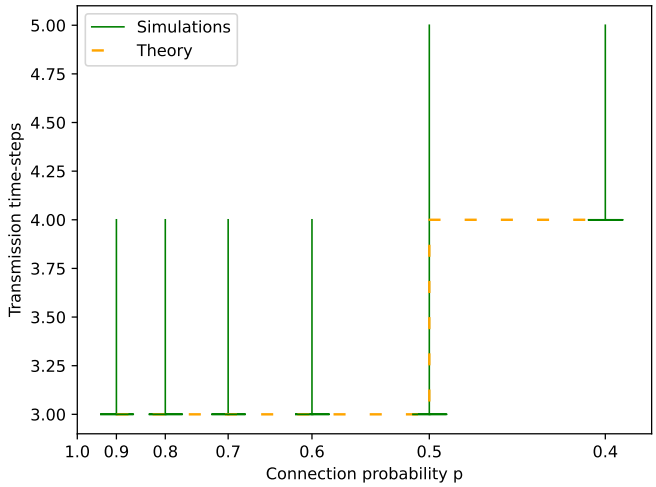


Figure 6.3: Graphic summarising 10,000 Monte Carlo simulations of Algorithm 2, in which the agents achieve allcast by broadcasting random linear combinations of previously received messages. The network contains $n = 256$ agents, and the edge connection probability p is plotted on a reciprocal scale. The y-axis shows the number of timesteps after which *every* agent may decode *every* message. The asymptotic bound given in Theorem 5.5 is plotted with a broken line for comparison. Notice how well the simulations agree with the theory.

simulations for any value of p. And since there is one partial linear combination per subset, the required number of timesteps jumps from 4 to 3 after p reaches 0.5.

Figure 6.5 shows simulations and predictions for increasing values of n. The overlap with the asymptotic bound (plotted with broken line) and the diminishing whiskers demonstrates that the performance of the method converges to predictions quickly.

6.4 Discussion

Recall from Theorem 5.2 that allcast on this graph model is not possible in fewer than $\lceil \frac{1}{p} \rceil$ time-steps, with high probability. Algorithm 2 can achieve this in one additional time-step, which suggests that it is close to optimal. This is far superior to Algorithm 1 (the *random message forwarding* method of Chapter 5), which requires a number of time-steps which increases logarithmically with the network size. This will allow the algorithms to be applied to much greater networks in practice, due to the decreased scaling in delay.

The price paid for this is a loss of decentralisation. Other than a stopping con-

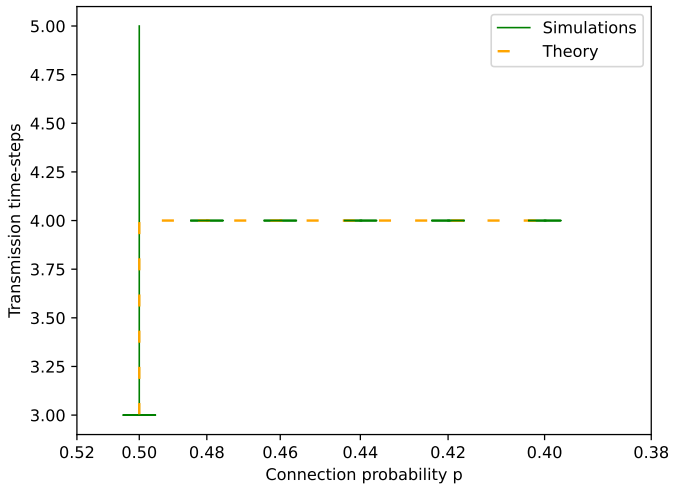


Figure 6.4: A similar graphic to Figure 6.3, with additional values of p in the region of $p = 0.5$. Notice the sharp transition, which is due to the design of the algorithm. The number of subsets the agents are partitioned into increases from 1 to 2 as p drops below 0.5. Hence, an extra partial linear combination must be sent by each agent.

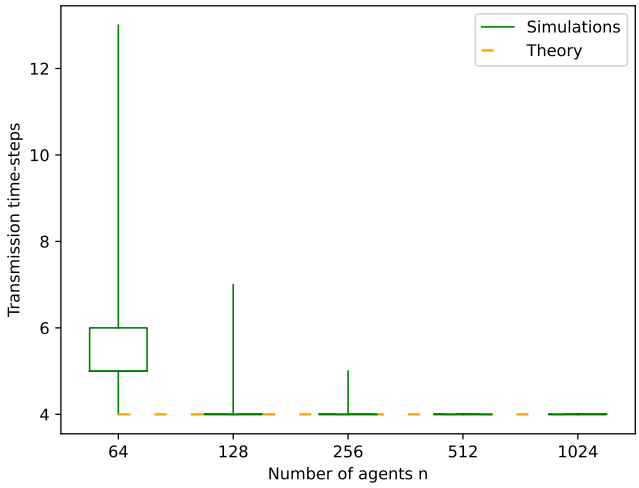


Figure 6.5: Graphic summarising 10,000 Monte Carlo simulations of Algorithm 2, in which the agents achieve allcast by broadcasting random linear combinations of previously received messages. The graph consists of n agents, pairs of which are connected with probability $p = 0.4$. The asymptotic bound given in Theorem 5.5 is plotted with a broken line for comparison. The simulation results show fast convergence with n to the asymptotic bound.

dition (which would also be required by this method), Algorithm 1 is completely distributed, and requires no cooperation between agents to implement. By comparison, the RLNC method introduced in this chapter requires knowledge of the size of the graph and edge connection probability, and a negotiation process to mutually agree a partition of the agents ahead of communication time. Another solution to the problem based on RLNC will be introduced in the next chapter, which does not place these requirements on the agents, and can be fully distributed in its implementation (given a labelling of the agents).

Whilst (according to both Theorem 6.1 and the simulations) the performance is only one timestep from optimal, it is interesting that optimal performance is never observed in simulations. In is interesting that, in ten thousand simulations, not one realisation fell below the asymptotic upper bound. You would think that as p approached 1, since there are a great deal of excess messages received, that decoding may be possible in one fewer round. We may speculate that this is because many of these extra messages will be partial linear combinations, which can only assist decoding of a single subset of messages. Although there are $O(n)$ of these in excess for each subset w.h.p, perhaps in practice these are less useful that full linear combinations. It would also be interesting to see whether increasing the density of the matrices would eliminate this phenomenon; this is left as further work. The method in the next chapter does not exhibit this behaviour.

Chapter 7

Network coding over random graphs: a decentralised method

Chapter 6 introduced Algorithm 2, an allcast algorithm based on Random Linear Network Coding (RLNC), in which the nodes are partitioned and a method reminiscent of chunking is applied. Whilst the algorithm is close to optimal in terms of the number of time-steps required for its completion, the setup required before communication commences complicates its deployment, and is an impediment to a fully decentralised implementation. Moreover, this limits its applicability to networks with a more dynamic topology, as the partition would have to be frequently re-negotiated. Solutions to both of these issues are highly desirable in many applications, such as vehicular communication.

This chapter introduces a fully decentralised algorithm, for which the agents require no knowledge of the graph. This comes at the expense of an in-depth study of the ranks of sparse random matrices.

7.1 Random linear network coding

In contrast to Algorithm 2, we now drop the idea of partitioning the agents, noting that this restriction on coding was introduced solely for ease of analysis. As before, in the first time-step, each agent broadcasts its own message. In subsequent time-steps, the agents compute random linear combinations of their neighbours messages (received in the first round), over the finite field $\mathbb{F}_q$. Again, as the analysis is not sensitive to the choice of q, we choose to code over the binary field $\mathbb{F}_2$. Coding over larger fields has the disadvantage of higher decoding complexity and an increased overhead caused by sharing coefficients [50]. The agents broadcast this linear combination, along with the coefficients used. To enable decoding, the coefficients of

all linear combinations are included as packet headers. This assumes that a unique labelling of each node can be arranged ahead of time. Algorithm 3, described below, is parametrised by a constant $\beta > 0$. The parameter does not depend on n, the system size.

Algorithm 3 Decentralised network coding algorithm for allcast on Erdős-Rényi random graphs.

1: In the first round, each agent broadcasts its own message. It also stores each message received in the first time-step in a buffer. Let N_v^{in} denote the set of messages received by agent v in the first time-step, and let $d_v^{\text{in}} = |N_v^{\text{in}}|$.

2: In each subsequent time-step, each agent broadcasts a random linear combination, over $\mathbb{F}_2$, of messages it received in the first round, computed as follows. Let $X_{vw}(t)$ denote the coefficient assigned by agent v to agent w's packet in round t. Then, $X_{vw}(t)$ are mutually independent Bernoulli random variables, and

$$\mathbb{P}(X_{vw}(t) = 1) = \begin{cases} (\beta \log d_v^{\text{in}})/d_v^{\text{in}}, & w \in \mathrm{N}^{\mathrm{in_v}}, \\ 0, & \text{otherwise.} \end{cases}$$

3: At the end of each time-step, each agent attempts to decode all of the messages, based on the linear combinations it has received in all previous time-steps. It succeeds when it has received $n - 1$ linearly independent vectors of coefficients.

4: If a node successfully decodes all packets after round t, then it terminates transmission after round $t + 3$.

The termination condition for the algorithm appears arbitrary, but is justified by the theoretical analysis presented below. The analysis shows that with high probability, for a suitable choice of β, every node will have received $n - 1$ linearly independent combinations after $\lceil 1/p \rceil + 2$ rounds; hence, allcast is successful after this many rounds. Conversely, by the same reasoning as in the proof of Theorem 5.2, no node will have received $n - 1$ linearly independent combinations in fewer than $\lfloor 1/p \rfloor$ rounds, with high probability. Hence, by terminating three rounds after it succeeded in decoding all packets, no node terminates prematurely, before all other nodes have successfully decoded all packets.

Algorithm 3 is fully distributed, and requires no knowledge about the system on the part of individual agents or nodes. In particular, agents do not need to know n or p.

Again, once each node has received a linearly independent set of $n - 1$ random linear combinations, the original messages may be obtained by solving the resulting linear system (for instance, by Gaussian elimination). Theorem 7.2 gives an asymp-

totic bound on the number of time-steps until this is possible for all nodes. Before stating and proving the theorem, we introduce an important definition regarding linear dependencies of matrices.

Definition 7.1. Let $m, n \in \mathbb{N}$, let $\boldsymbol{A}$ be an $n \times m$ matrix with entries in $\mathbb{F}_2$, and $\boldsymbol{x} \in \mathbb{F}_2^n$, $\boldsymbol{x} \neq 0$. Then $\boldsymbol{x}$ is said to be a *linear dependency of the columns of* $\boldsymbol{A}$ exactly when $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$. We define $\lambda(\boldsymbol{A}) = |\{\boldsymbol{x} \in \mathbb{F}_2^n : (\boldsymbol{x} \neq \boldsymbol{0}) \wedge (\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0})\}|$ to be the total number of linear dependencies of the columns of $\boldsymbol{A}$.

Theorem 7.2. *Fix $\tau \in \mathbb{R}^+$, $\beta \geq \frac{5}{(1-\tau)^3(1+Mp)}$, and suppose all agents employ Algorithm 3. Suppose the modelling assumptions of Section 5.1 are satisfied. Let $M \in \mathbb{N}$ and X_M denote the event that, after $\lceil \frac{1}{p} \rceil + M + 1$ rounds, every node has a full rank system of linear equations, and hence may decode all messages. Then for all $M \in \mathbb{N}$,*

$$\lim_{n \longrightarrow \infty} \mathbb{P}(X_M^{\mathrm{c}}) = 0.$$

Remark. It is clear to see from Theorem 7.3 that the expected number of linear dependencies is monotone decreasing for $\pi \in (\beta \frac{\log(n)}{np}, \frac{1}{2})$, and hence the Theorem 7.2 holds for all π in this interval.

Remark. Since the result holds for all $M \in \mathbb{N}$, in particular it holds for $M = 1$, and decoding is hence possible for all nodes with high probability after $\lceil \frac{1}{p} \rceil + 2$ rounds. Notice that larger M allows a larger choice of τ and hence faster convergence; the upper bound on τ is simply ensuring that each matrix $\boldsymbol{A}_i$ has more rows than columns by at least some constant multiple of n.

We begin with a proof of Theorem 7.2, which follows from the results in the next two sections.

Proof. Suppose each node discards the messages received in the first round as soon as transmissions are complete, and does not use them to aid in decoding. Note that this assumption leads to an upper bound on $\mathbb{P}(X_M^{\mathrm{c}})$, as the discarded packets may aid decoding. The interested reader may easily show that these discarded packets may be used to allow decoding after one fewer round if π is suitable chosen (and larger).

Let $\tau \in \left(0, \frac{Mp-1}{Mp}\right) \subset (0, 1)$, let B_k denote the event that node k has at least $\lceil (1-\tau)np \rceil$ in-neighbours. Note that by Lemma 2.10, $\mathbb{P}(B_i^{\mathrm{c}}) \leq \exp(-nD((1-\tau)p; p))$. We now assume that each node selects $\lceil (1-\tau)np \rceil$ of its in-neighbours from which to accept messages, and discards all messages from any other neighbours. Note that this assumption can only increase the $\mathbb{P}(X_M^{\mathrm{c}})$, as the discarded packets may aid decoding.

Let $\boldsymbol{A}_k$ be a matrix whose rows are formed by the coefficients of the remaining linear combinations received by node k after $\lceil \frac{1}{p} \rceil + M + 1$ time-steps. Suppose w.l.o.g that the rows of $\boldsymbol{A}_k$ are ordered by the node which sent them. We may label sets of rows corresponding to messages received from each node j to be sub-matrices $\boldsymbol{R}_j$, each of which is $\rho \times n$, where $\rho := \lceil \frac{1}{p} \rceil + M + 1$ is the number of time-steps and hence messages received from each in-neighbour.

Label the columns of $\boldsymbol{R}_j$ by $\boldsymbol{C}_{i,\boldsymbol{R}_j}$, which we refer to as the *subcolumns* of $\boldsymbol{R}_j$. Entries in the subcolumns $\boldsymbol{C}_{i,\boldsymbol{R}_j}$ represent the inclusion of the message of node i in the linear combinations formed by the in-neighbour corresponding to j. If node i is not an in-neighbour of j, which occurs with probability $1-p$, then j cannot possibly include this message in any linear combination, and all entries of $\boldsymbol{C}_{i,\boldsymbol{R}_j}$ are forced equal to zero. Otherwise, the entries are chosen i.i.d according to a Bernoulli(π_j) distribution, where

$$\begin{aligned}
\pi_j &= \beta \frac{\log(d_v^{\text{in}})}{d_v^{\text{in}}} \\
&\geq \beta \frac{\log(\lceil (1-\tau)np \rceil)}{\lceil (1-\tau)np \rceil} \\
&\geq \beta \frac{\log((1-\tau)np)}{(1-\tau)np} \\
&\geq \frac{5}{(1-\tau)^3(1+Mp)} \frac{\log((1-\tau)np)}{np} \\
&\geq \frac{5}{(1-\tau)^2(1+Mp)} \frac{\log(n)}{np},
\end{aligned}$$

for n large enough. Notice that $\boldsymbol{A}_k$ is a random matrix *exactly* of the form detailed in Theorem 7.3, with $\pi_j \in [\frac{5}{(1-\tau)^2(1+Mp)} \frac{\log(n)}{np}, 1]$. Thus, by Theorem 7.4, $\lim_{n \longrightarrow \infty} n\mathbb{E}(\lambda(\boldsymbol{A}_k)) = 0$. By a simple application of Markov's inequality, the probability of $\boldsymbol{A}_k$ not being full rank given B_k is $o(n^{-1})$. Now since Algorithm 3 requires more than $\lceil \frac{1}{p} \rceil + M + 1$ time-steps if and only if at least one of the nodes does not have a full rank decoding matrix, we have

$$\begin{aligned}
\mathbb{P}(X_M^{\text{c}}) &\leq \mathbb{P}\big(\cup_{i=k}^{n} \{\lambda(\boldsymbol{A}_k) \geq 1\}\big) \\
&\leq n\mathbb{P}\big(\lambda(\boldsymbol{A}_1) \geq 1\big) \\
&= n\Big(\mathbb{P}\big(\lambda(\boldsymbol{A}_1) \geq 1 | B_1\big)\mathbb{P}(B_1) + \mathbb{P}\big(\lambda(\boldsymbol{A}_1) \geq 1 | B_1^{\text{c}}\big)\mathbb{P}(B_1^{\text{c}})\Big) \\
&\leq n\mathbb{P}\big(\lambda(\boldsymbol{A}_1) \geq 1 | B_1\big) + n\exp(-nD((1-\tau)p; p)),
\end{aligned}$$

where the first inequality follows since we may have discarded useful messages, and the second from the Union bound. Applying our observation that $\mathbb{P}\big(\lambda(\boldsymbol{A}_1) \geq 1 | B_1\big) = o(n^{-1})$, we obtain

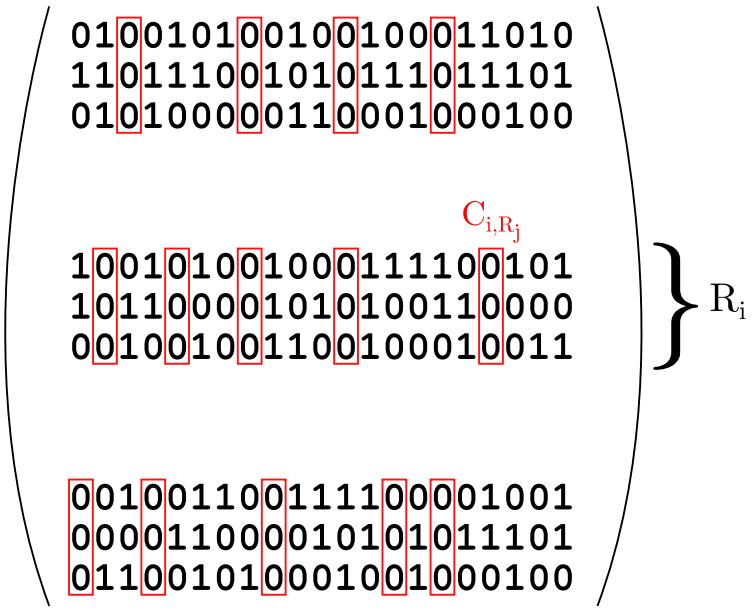


Figure 7.1: Graphic demonstrating the challenges of proving Theorem 7.3. Notice the red boxes, showing subcolumns of the matrix which are forced equal to zero, and whose entries are not independent.

$$\lim_{n\longrightarrow\infty} \mathbb{P}(X_M^{\mathrm{c}}) = 0. \qquad \square$$

7.2 Expected rank of random matrices

We begin with an exact expression for the expected number of linear dependencies (defined below) of the columns of the matrices $\boldsymbol{A}_k$, whose rows are formed from the coefficients of random linear combinations received by decoding time by a particular transceiver node i. Since the following two sections simply concern random matrices, we drop the subscript k from our notation for clarity.

The main challenge in constructing this proof is illustrated in Figure 7.1. The red boxes show sections of the matrix, which will be defined later as *subcolumns*, which are forced to be zero. These are groups of entries of the matrix which are not independent. The diagram also illustrates the notation defined and used in the proof of Theorem 7.3.

Theorem 7.3. *Let $p \in (0,1]$, $M, n \in \mathbb{N}$, let $\tau \in \left(0, \frac{Mp}{Mp+1}\right)$, let $\rho = \left\lceil \frac{1}{p} \right\rceil + M$ and let $\pi_1, \ldots, \pi_\rho \in (0,1]$. Let $\boldsymbol{A}$ be an $n \times \lceil (1-\tau)np \rceil \rho$ matrix with entries in $\mathbb{F}_2$. It is easy to show that $\lceil (1-\tau)np \rceil \rho \geq (1+c)n$ for some constant $c \in \mathbb{R}^+$, that is, the matrix has at least cn more rows than columns. Label each row of $\boldsymbol{A}$ by $\boldsymbol{r}_j$. Partition the rows into submatrices $\boldsymbol{R}_j = (\boldsymbol{r}_{j\rho}^T, \ldots, \boldsymbol{r}_{(j+1)\rho-1}^T)^T$, and define the* subcolumns $\boldsymbol{C}_{i,\boldsymbol{R}_j}$

to be the ith column of $\boldsymbol{R}_j$.

Suppose $\boldsymbol{A}$ *is chosen at random as follows. Each subcolumn* $\boldsymbol{C}_{i,\boldsymbol{R}_j}$ *of* $\boldsymbol{A}$ *is chosen to be the zero vector i.i.d with probability* $(1-p)$. *Otherwise, each of its entries are chosen i.i.d according to the* Bernoulli(π_j) *distribution. The submatrices are independent. Then*

$$\mathbb{E}(\lambda(\boldsymbol{A})) = \sum_{k=1}^{n} \binom{n}{k} 2^{-\lceil (1-\tau)np \rceil \rho} \prod_{j=1}^{\lceil (1-\tau)np \rceil} \sum_{s=0}^{k} \binom{k}{s} p^s (1-p)^{k-s} \big(1+(1-2\pi_j)^s\big)^{\rho}. \quad (7.1)$$

Moreover, if there exists $\pi' \in (0,1]$ *such that* $\pi \in [\pi', 1]$, $\forall j \in 1, \ldots, \rho$, *then*

$$\mathbb{E}(\lambda(\boldsymbol{A})) \leq \sum_{k=1}^{n} \binom{n}{k} 2^{-\lceil (1-\tau)np \rceil \rho} \left(\sum_{s=0}^{k} \binom{k}{s} p^s (1-p)^{k-s} \big(1+(1-2\pi')^s\big)^{\rho} \right)^{\lceil (1-\tau)np \rceil}. \quad (7.2)$$

Remark. Note that the π_j here have no relationship with any agent (or agent j in particular); this is simply a theorem regarding random matrices. However, when applied to the proof of Theorem 7.2, they will represent the encoding densities of the agents, but in no particular order. For instance, π_j need not correspond to agent j, as the matrix in question may belong to an agent which is not in the out-neighbourhood of j.

Proof. Let $\boldsymbol{x}_k \in \mathbb{F}^n$ be a column vector with exactly k non-zero entries. We begin by evaluating the probability that $\boldsymbol{x}_k$ is a linear dependency of the columns of $\boldsymbol{A}$. Since the columns of $\boldsymbol{A}$ are i.i.d, we may assume w.l.o.g that the first k entries of $\boldsymbol{x}_k$ are equal to 1, and that the rest are zero. Notice that $\boldsymbol{x}_k$ is a linear dependency of the columns of $\boldsymbol{A}$ if and only if $\boldsymbol{x}_k$ is a linear dependency of the columns of every submatrix $\boldsymbol{R}_j$. That is

$$\boldsymbol{A}\boldsymbol{x}_k = 0 \iff (\forall j \in \{1, \ldots, np\}) \Big(\sum_{i=0}^{k} \boldsymbol{C}_{i,\boldsymbol{R}_j} = 0 \Big),$$

noting that $\boldsymbol{R}_j \boldsymbol{x} = \sum_{i=0}^{k} \boldsymbol{C}_{i,\boldsymbol{R}_j}$ since only the first k entries of $\boldsymbol{x}_k$ are non-zero.

Notice that $\boldsymbol{x}_k$ is a linear dependency of the columns of each submatrix *independently*, and we will continue by evaluating this probability. Fix j. Since we are evaluating the probability that the sum of the first k subcolumns is zero, we may assume w.l.o.g that the first S subcolumns $\boldsymbol{C}_{1,\boldsymbol{R}_j}, \ldots, \boldsymbol{C}_{S,\boldsymbol{R}_j}$, $s \leq k$ are not "forced" to equal the zero vector, that the entries of these columns are chosen according to a Bernoulli(π) distribution (which permits the subcolumns to be zero by chance), and that the following $k - S$ subcolumns are forced equal to zero. Clearly, $S \sim \text{Bin}(k, p)$, since the subcolumns are not forced to be zero i.i.d with probability p.

Let $P_s = \mathbb{P}(\sum_{i=1}^{s}(\boldsymbol{R}_j)_{1,i} = 0 \,|\, S = s)$. Following the proof of Theorem 3.3 from [25], we have

$$P_0 = 1,\ P_s = P_{s-1}(1-\pi_j) + (1-P_{s-1})\pi_j.$$

Using the substitution $Q_s = P_s = \frac{1}{2}$, we have $Q_0 = \frac{1}{2}$ and

$$\begin{aligned} Q_s &= (Q_{s-1} + \tfrac{1}{2})(1-\pi_j) + (\tfrac{1}{2} - Q_{s-1})\pi_j - \tfrac{1}{2} \\ &= Q_{s-1} + \tfrac{1}{2} - 2\pi_j Q_{s-1} - \tfrac{1}{2} \\ &= Q_{s-1}(1-2\pi_j). \end{aligned}$$

Hence $Q_s = \frac{1}{2}(1-2\pi_j)^s \implies P_s = \frac{1}{2}(1-2\pi_j)^s + \frac{1}{2}$. We have

$$\begin{aligned} \mathbb{P}\big(\textstyle\sum_{i=1}^{k}\boldsymbol{C}_{i,\boldsymbol{R}_j} = 0\big) &= \sum_{s=0}^{k}\mathbb{P}(\textstyle\sum_{i=1}^{k}\boldsymbol{C}_{i,\boldsymbol{R}_j} = 0 | S = s)\mathbb{P}(S = s) \\ &= \sum_{s=0}^{k}\mathbb{P}(\mathrm{Bin}(k,p) = s)\big(\tfrac{1}{2} + \tfrac{1}{2}(1-2\pi_j)^s\big)^{\rho}, \end{aligned}$$

by the independence of the rows of the submatrix. Now, by the independence of the submatrices, we see that

$$\mathbb{P}(\boldsymbol{A}\boldsymbol{x}_k = 0) = \prod_{j=1}^{\lceil(1-\tau)np\rceil}\sum_{s=0}^{k}\mathbb{P}(\mathrm{Bin}(k,p) = s)\big(\tfrac{1}{2} + \tfrac{1}{2}(1-2\pi_j)^s\big)^{\rho}.$$

Summing over k and multiplying by the number of k-vectors gives

$$\begin{aligned} \mathbb{E}(\lambda(\boldsymbol{A})) &= \sum_{k=1}^{n}\binom{n}{k}\prod_{j=1}^{\lceil(1-\tau)np\rceil}\sum_{s=0}^{k}\binom{k}{s}p^s(1-p)^{k-s}\big(\tfrac{1}{2} + \tfrac{1}{2}\big(1-2\pi_j\big)^s\big)^{\rho} \qquad (7.3) \\ &= \sum_{k=1}^{n}\binom{n}{k}2^{-\lceil(1-\tau)np\rceil\rho}\prod_{j=1}^{\lceil(1-\tau)np\rceil}\sum_{s=0}^{k}\binom{k}{s}p^s(1-p)^{k-s}\big(1 + (1-2\pi_j)^s\big)^{\rho}. \end{aligned}$$

The final claim of the Theorem follows since the terms of the product are monotone decreasing in π_j. □

7.3 Upper bound for random matrix singularity probability

We are now interested in an asymptotic regime in which n tends to infinity, while $\pi_j \geq \pi := \frac{\beta \log n}{np}$, for all j and a fixed constant $\beta > 0$, and most importantly for which $\lambda(A) = o(n^{-1})$. We do not make the dependence of π on n, p and β explicit

in the notation; thus, p and β are fixed, while n tends to infinity and π tends to zero. For convenience, we introduce the following notation:

$$\mathbb{E}(\lambda(A)) = \sum_{k=1}^{n} \binom{n}{k} 2^{-\lceil (1-\tau)np \rceil \rho} f(k,p,\pi)^{\lceil (1-\tau)np \rceil}, \tag{7.4}$$

where $n \in \mathbb{N}$, $p \in (0,1)$, $\pi \in (0,1/2)$, and

$$f(k,p,\pi) = \sum_{s=0}^{k} \binom{k}{s} p^s (1-p)^{k-s} \big(1 + (1-2\pi)^s\big)^{\rho}, \ k = 0,1,\ldots,n. \tag{7.5}$$

Theorem 7.4. *Let $\boldsymbol{A}$ be a random matrix as defined in Theorem 7.3, and recall $\mathbb{E}(\lambda(\boldsymbol{A}))$ as stated in (7.4). Fix $M \in \mathbb{N}$, $p > 0$, $\beta \geq \frac{5}{(1-\tau)^2(1+Mp)}$ and, for $n \in \mathbb{N}$, set $\pi_j \geq \frac{\beta \log n}{np}$. Then*

$$\lim_{n \longrightarrow \infty} n\mathbb{E}(\lambda(\boldsymbol{A})) = 0$$

Moreover, the probability that $\boldsymbol{A}$ is singular is $o(n^{-1})$.

The proof of the second claim may be obtained by applying Markov's inequality to the first: the proof which will proceed through a sequence of lemmas.

Lemma 7.5. *Let $x \in (0,1/2)$. Then,*

$$(1-2x)^s \leq 1 - xs \quad \forall\, 0 < s < \frac{\log 2}{-\log(1-2x)}.$$

Remark. We will be interested in the above bound for x much smaller than 1, for which the bound on s in the lemma statement is approximately $\frac{\log 2}{2x}$, which is a constant multiple (not depending on n) of $1/x$.

Proof. Fix x and define $g(s) = (1-2x)^s - 1 + xs$. Then, $g(0) = 0$ and

$$g'(s) = (1-2x)^s \log(1-2x) + x,$$

which is negative for all $s \in (0, \frac{\log 2}{-\log(1-2x)})$. This completes the proof. □

Lemma 7.6. *Let π be as in the statement of Theorem 7.4 and let $k^* = \lfloor \frac{\alpha n}{\log n} \rfloor$, where $\alpha > 0$ is a fixed constant. If α is sufficiently small, then $f(k,p,\pi) \leq 2^{\rho} \exp(-(1 + Mp)\frac{\pi k}{4})$ for $k = 0,1,\ldots,k^*$.*

Proof. For small enough α, and $s \leq k \leq k^*$, we have by Lemma 7.5 that $(1-2\pi)^s \leq 1 - \pi s$. Hence, for $k \leq k^*$,

$$\begin{aligned} f(k,p,\pi) &\leq \sum_{s=0}^{k} \binom{k}{s} p^s (1-p)^{k-s} (2-\pi s)^{\rho} \\ &\leq 2^{\rho} \sum_{s=0}^{k} \binom{k}{s} p^s (1-p)^{k-s} \Big(1 - \rho \frac{\pi s}{4}\Big) \end{aligned}$$

$$= 2^\rho\Big(1 - kp\rho\frac{\pi}{4}\Big)$$
$$\leq 2^\rho \exp(-p\rho\frac{\pi k}{4}),$$

where we have used Lemma 7.5 once more to obtain the second inequality, the equality follows by recognising the expectation of a binomial random variable, and the final inequality follows from the inequality $e^{-x} \geq 1-x$. The result of the lemma follows since $p\rho \geq 1 + Mp$. □

Lemma 7.7. *Let π and k^* be as in the statement of Lemma 7.6. Then, for $k = k^* + 1, \ldots, n$,*

$$f(k,p,\pi) \leq \left(1 + \exp(-kp\pi) + 2\exp\Big(-\tfrac{k}{\rho}D\big(\tfrac{p}{2};p\big)\Big)\right)^\rho.$$

Proof. Decompose $f(k,p,\pi)$ as

$$f(k,p,\pi) = \sum_{s=0}^{\lceil kp/2\rceil - 1}\binom{k}{s}p^s(1-p)^{k-s}\big(1+(1-2\pi)^s\big)^\rho + \sum_{s=\lceil kp/2\rceil}^{k}\binom{k}{s}p^s(1-p)^{k-s}\big(1+(1-2\pi)^s\big)^\rho.$$

Since $(1-2\pi)^s$ is a decreasing function of s, the first sum on the right hand side above is bounded by $2^\rho\sum_{s=0}^{\lceil kp/2\rceil-1}\binom{k}{s}p^s(1-p)^{k-s}$; using Lemma 2.10, this is no bigger than $2^\rho e^{-kD(p/2;p)}$. Likewise, the second sum on the right hand side is bounded by

$$\big(1+(1-2\pi)^{kp/2}\big)^\rho \sum_{s=\lceil kp/2\rceil}^{k}\binom{k}{s}p^s(1-p)^{k-s},$$

which is no bigger than $\big(1+(1-2\pi)^{kp/2}\big)^\rho$. Combining these two bounds yields

$$f(k,p,\pi) \leq 2^\rho\Big(\exp\Big(-kD\Big(\frac{p}{2};p\Big)\Big) + \Big(\frac{1+(1-2\pi)^{kp/2}}{2}\Big)^\rho\Big).$$

Next, observe that $(a+b)^\rho \geq a^\rho + b^\rho$ for all $a, b \geq 0$ and all $\rho \geq 1$; this follows from the fact that $(1+x)^\rho \geq 1 + x^\rho$ for all $x \geq 0$ and $\rho \geq 1$, which can be demonstrated using elementary calculus. Hence, we obtain that

$$f(k,p,\pi) \leq 2^\rho\Big(\exp\Big(-\tfrac{k}{\rho}D\Big(\frac{p}{2};p\Big)\Big) + \frac{1+(1-2\pi)^{kp/2}}{2}\Big)^\rho.$$

The claim of the lemma now follows from the inequality $(1-x)^\alpha \leq e^{-\alpha x}$, which holds for all $\alpha > 0$.

Lemma 7.8. *Let $\alpha, \beta > 0$ be given. Let $\pi = \beta\frac{\log n}{np}$ and let $k^* = \lfloor \alpha n/\log n\rfloor$. Fix $\delta > 0$. Then, for all n sufficiently large and $k \in \{k^*+1, \ldots, n\}$, we have*

$$f(k,p,\pi)^{\lceil(1-\tau)np\rceil} \leq (1+\delta)[1+\exp(-kp\pi)]^{\lceil(1-\tau)np\rceil\rho}.$$

Proof. We have by Lemma 7.7 that

$$f(k,p,\pi)^{\lceil(1-\tau)np\rceil} \leq [1+\exp(-kp\pi)]^{\lceil(1+\tau)np\rceil\rho}[1+2\exp(-\tfrac{k}{\rho}D(p/2;p))]^{\lceil(1-\tau)np\rceil\rho}. \tag{7.6}$$

Let $\lambda = \rho^{-1}D(p/2;p)$. Then, λ is a positive constant that does not depend on n, and we can write

$$\begin{aligned}[1+2\exp(-kpD(p/2;p))]^{\lceil(1-\tau)np\rceil\rho} &= (1+2e^{-\lambda k})^{\lceil(1-\tau)np\rceil\rho} \\ &\leq \exp(2\lceil(1-\tau)np\rceil\rho e^{-\lambda k}) \\ &\leq \exp\Big(2\lceil(1-\tau)np\rceil\rho\exp\Big(-\frac{\lambda\alpha n}{\log n}\Big)\Big).\end{aligned}$$

Substituting this into (7.6), and noting that $2n\exp(-\lambda\alpha n/\log n)$ tends to zero as n tends to infinity, we obtain the claim of the lemma. □

Next, we decompose the sum in the definition of $\mathbb{E}\big(\lambda(\boldsymbol{A})\big)$ in (7.4) and bound each of the pieces. Firstly, we have:

Lemma 7.9. *Fix $p>0$, $\tau\in\mathbb{R}^+$, $\beta=\frac{16}{(1-\tau)^2(1+Mp)}$, and let $\pi=\beta\frac{\log n}{np}$. Let $\alpha>0$ be sufficiently small that the conclusion of Lemma 7.6 holds with $k^*=\lceil\alpha n/\log n\rceil$, for all n sufficiently large. Then,*

$$\sum_{k=1}^{k^*}\binom{n}{k}2^{\lceil(1-\tau)np\rceil\rho}f(k,p,\pi)^{\lceil(1-\tau)np\rceil} = O\big(n^{-(1+\tau)^{-1}}\big).$$

Proof. Using Lemma 7.6 and the inequality $\binom{n}{k}\leq n^k/k!$, we have

$$\begin{aligned}\sum_{k=1}^{k^*}\binom{n}{k}2^{-\lceil(1-\tau)np\rceil\rho}f(k,p,\pi)^{\lceil(1-\tau)np\rceil} &\leq \sum_{k=1}^{k^*}\frac{1}{k!}\exp\Big(k\log n-(1+Mp)\frac{\pi k\lceil(1-\tau)np\rceil}{4}\Big) \\ &\leq \sum_{k=1}^{k^*}\frac{1}{k!}\exp\Big(k\log n-(1+Mp)\frac{(1-\tau)\pi knp}{4}\Big) \\ &\leq \sum_{k=1}^{\infty}\frac{1}{k!}\exp\Big(-\frac{\beta(1+Mp)(1-\tau)-4}{4}k\log n\Big) \\ &= e^{n^{-(\beta(1+Mp)(1-\tau)-4)/4}}-1 \\ &= \exp(n^{-(1-\tau)^{-1}})-1.\end{aligned}$$

The claim of the lemma follows by a simple application of Taylor's theorem. □

Lemma 7.10. *Let $\alpha,\beta>0$ be given, let π and k^* be defined as in Lemma 7.8. Fix $\epsilon>0$ sufficiently small that $e^{\epsilon}(1+e^{-\alpha\beta})<2$. Then, there is a unique $\gamma=\gamma_\epsilon$ in $(0,1/2)$ such that*

$$D(\gamma;1/2)=\log(1+e^{-\alpha\beta})+\epsilon.$$

Moreover,

$$\lim_{n\to\infty} n \sum_{k=k^*+1}^{\lfloor \gamma n \rfloor} \binom{n}{k} 2^{-\lceil (1-\tau)np \rceil \rho} f(k,p,\pi)^{\lceil (1-\tau)np \rceil} = 0$$

Proof. Since $D(x;1/2)$ decreases monotonically from $\log 2$ at $x = 0$ to 0 at $x = 1/2$, the first claim of the lemma is straightforward. Next, since $k^* p\pi \geq \alpha\beta$, we see from Lemma 7.8 and the definition of γ that

$$f(k,p,\pi)^{\lceil (1-\tau)np \rceil} \leq (1+\delta)[1+e^{-\alpha\beta}]^{\lceil (1-\tau)np \rceil \rho} \leq (1+\delta)e^{-\epsilon \lceil (1-\tau)np \rceil \rho} e^{\lceil (1-\tau)np \rceil \rho D(\gamma;1/2)},$$

for all $k \in \{k^*+1, \ldots, \lfloor \gamma n \rfloor\}$, and arbitrarily small $\delta > 0$.

Thus, using the inequality $\log(\binom{n}{k} 2^{-n}) \leq -nD(k/n;1/2)$, which follows easily from Stirling's formula, we obtain for all $k \in \{k^*+1, \ldots, \lfloor \gamma n \rfloor\}$ and all n sufficiently large, that

$$\begin{aligned} \binom{n}{k} 2^{-\lceil (1-\tau)np \rceil \rho} f(k,p,\pi)^{\lceil (1-\tau)np \rceil} &\leq (1+\delta) e^{\lceil (1-\tau)np \rceil \rho [D(\gamma;1/2) - D(k/n;1/2) - \epsilon]} \\ &\leq (1+\delta) e^{-\epsilon \lceil (1-\tau)np \rceil \rho}. \end{aligned}$$

We have used the monotonicity of $D(\cdot;1/2)$ on $(0,1/2)$ to obtain the last inequality. It now follows that

$$n \sum_{k=k^*+1}^{\lfloor \gamma n \rfloor} \binom{n}{k} 2^{\lceil (1-\tau)np \rceil \rho} f(k,p,\pi)^{\lceil (1-\tau)np \rceil} \leq (1+\delta)\gamma n^2 e^{-\epsilon \lceil (1-\tau)np \rceil \rho}.$$

As γ, δ and ϵ are positive constants which don't depend on n, the lemma is proved by letting n tend to infinity. □

Lemma 7.11. *Let γ be defined as in Lemma 7.10, and τ, c as defined in Theorem 7.3. Then,*

$$\lim_{n\to\infty} n \sum_{k=\lfloor \gamma n \rfloor + 1}^{\lfloor n/\beta \rfloor} \binom{n}{k} 2^{-\lceil (1-\tau)np \rceil \rho} f(k,p,\pi)^{\lceil (1-\tau)np \rceil} = 0,$$

where we define an empty sum to be zero.

Proof. If $\gamma \geq 1/\beta$, the lemma holds trivially since an empty sum is zero. Henceforth, we restrict attention to the case $\gamma < 1/\beta$. First, observe from Lemma 7.8 and the definition of π that, for all $k \geq \gamma n$,

$$\begin{aligned} f(k,p,\pi)^{\lceil (1-\tau)np \rceil} &\leq (1+\delta)[1+\exp(-kp\pi)]^{\lceil (1-\tau)np \rceil \rho} \\ &\leq (1+\delta)[1+n^{-\beta\gamma}]^{\lceil (1-\tau)np \rceil \rho} \\ &\leq (1+\delta)\exp(n^{1-\beta\gamma}(1+Mp)), \end{aligned}$$

since $np \geq \lceil (1-\tau)np \rceil$ for large enough n. Hence, using the bound $\log(\binom{n}{k}2^{-n}) \leq -nD(k/n;1/2)$ and the fact that $D(\cdot;1/2)$ is monotone decreasing on $[0,1/2]$, we obtain for all $k \in [\gamma n, n/\beta]$ that

$$\binom{n}{k}2^{-\lceil (1-\tau)np \rceil \rho} f(k,p,\pi)^{\lceil (1-\tau)np \rceil} \leq (1+\delta)2^{-nc}\exp(-n[D(1/\beta;1/2)-(1+Mp)n^{-\beta\gamma}]),$$

for some $c \in \mathbb{R}^+$. Summing over k and multiplying by n, we have

$$n\sum_{k=\lfloor \gamma n \rfloor + 1}^{\lfloor n/\beta \rfloor} \binom{n}{k}2^{-\lceil (1-\tau)np \rceil \rho} f(k,p,\pi)^{np} \leq (1+\delta)2^{-nc}n^2\exp(-n[D(1/\beta;1/2)-(1+Mp)n^{-\beta\gamma}]).$$

Since $D(1/\beta;1/2)$ is a positive constant, while $n^{-\beta\gamma}$ tends to zero as n tends to infinity, the claim of the lemma follows by taking limits as n tends to infinity on both sides of the inequality above. □

Lemma 7.12. *Let $\beta > 0$ be given, let $\pi = \beta \log n/np$, and let $\delta \in \mathbb{R}^+$. Let τ and c be defined as in Theorem 7.3. Then $\exists N \in \mathbb{N}$ such that for $n > N$,*

$$\sum_{k=\lfloor n/\beta+1 \rfloor}^{n} \binom{n}{k}2^{-\lceil (1-\tau)np \rceil \rho} f(k,p,\pi)^{\lceil (1-\tau)np \rceil} \leq (1+\delta)2^{-cn}e^{1+Mp}.$$

Proof. We see from Lemma 7.8 that, for all n sufficiently large and all $k > n/\beta$, we have

$$\begin{aligned} f(k,p,\pi)^{\lceil (1-\tau)np \rceil} &\leq (1+\delta)\big(1+e^{-np\pi/\beta}\big)^{\lceil (1-\tau)np \rceil \rho} \\ &\leq (1+\delta)\Big(1+\frac{1}{n}\Big)^{n(1+Mp)} \\ &\leq (1+\delta)e^{1+Mp}, \end{aligned}$$

where the second inequality follows since $np \geq \lceil (1-\tau)np \rceil$ for large enough n. Hence, for all n sufficiently large,

$$\begin{aligned} \sum_{k=\lfloor n/\beta+1 \rfloor}^{n} \binom{n}{k}2^{-\lceil (1-\tau)np \rceil \rho} f(k,p,\pi)^{\lceil (1-\tau)np \rceil} &\leq (1+\delta)2^{-cn}e^{1+Mp}\sum_{k=0}^{n}\binom{n}{k}2^{-n} \\ &\leq (1+\delta)2^{-cn}e^{1+Mp}. \quad \square \end{aligned}$$

Proof of Theorem 7.4. The proof is immediate from Lemmas 7.9, 7.10, 7.11 and 7.12. □

7.4 Simulation results

7.4.1 Static graphs

Whilst the bounds given in this section hold asymptotically, these results give no guarantee over the performance of the algorithm for small networks. A summary of

10,000 Monte-Carlo simulations are presented in this section for the case of static graphs (i.e $\alpha = 0$). These complement the rigorous analysis, showing that the bound converges quickly and is useful for small n. Figures 5.1 and 5.2 give a comparison of 10,000 simulations with the bound given in Theorem 5.5. Caps on the whiskers of the box plots throughout this section have not been plotted, because the true minima and maxima are 1 and ∞, respectively, and will not be attained in simulations. The whiskers do however illustrate the full range of simulation results. Also note that the box-plots have in some cases collapsed to lines, because the minimum, median, lower and upper quartile, and maximum, all coincide.

Figures 7.2 and 7.3 show the number of time-steps required by Algorithm 3 against n and p, on logarithmic and reciprocal x-axis scales, respectively. The algorithm's parameter was chosen as $\beta = 2$. Notice in Figure 7.2, the medians do approach the bound with n, with over 75% of observations at or below the bound for graphs of size $n \geq 256$. Similarly, for a moderate choice of graph size, Figure 7.3 shows that the majority of observations fell below the bound for all choices of edge connection probability p.

Notice that for large p, most of the simulations fall below the upper bound. Recall from the proof of Theorem 7.2 that it is assumed that the messages from the first round would not be used for decoding. This, and the large number of excess messages received in these situations are clearly helping the agents to decode the messages enough to save time-steps. It has already been mentioned that a bound of one fewer time-step can be proven if the density is suitably increased. These simulation results suggest that it may be possible for agents to either reduce their density somewhat, or perhaps under a different model for some agents to remain silent in some rounds, to reduce the number of excess messages. The simplest way to achieve this would be for agents to estimate p, and to refrain from transmitting (in any time-step) with probability $(\lceil \frac{1}{p} \rceil)^{-1} \cdot \frac{1}{p}$, effectively choosing a random subgraph with the correct number of edges and reducing excess transmissions. Better still would be to make this choice at every timestep (as the next section suggests that changing the topology can speed up dissemination). This would reduce the decentralised nature and perhaps robustness of the algorithm, however. Investigation of this idea is left as further work.

Next, we investigate the affect of the parameter β on the algorithm's performance. Theorem 7.2 suggests that this constant should be taken to be greater than 5, however simulation results show that to be excessive in practice. Figure 7.4 compares the performance of the network coding method for various values of n and β. The simulations suggest that for all n, improved performance can be obtained by

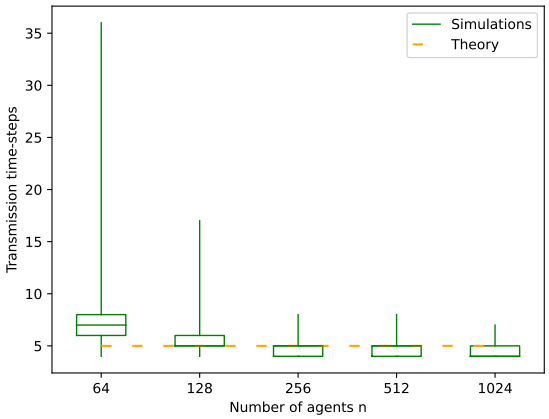


Figure 7.2: Graphic summarising 10,000 Monte Carlo simulations of Algorithm 3, in which the agents perform allcast by sharing random linear combinations of their neighbours messages at each timestep. The network consists of n agents, with edge connection probability $p = 0.4$. The box plots represent the number of time-steps required when $\beta = 2$. Note that the n-axis scale is logarithmic. The asymptotic bound given in Theorem 7.2, which is exceeded with probability vanishing in n, is plotted for comparison. Notice that the simulation results exceed this bound less frequently as n increases, suggesting that the bound converges quickly.

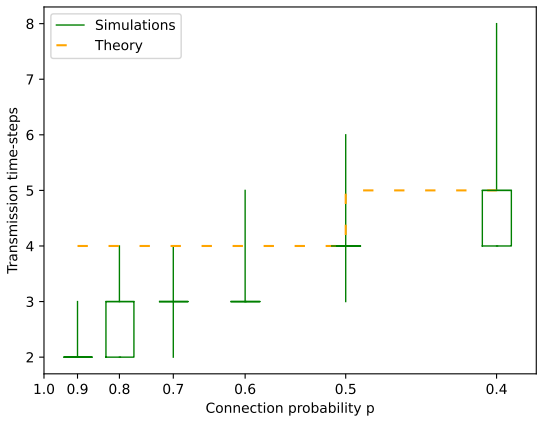
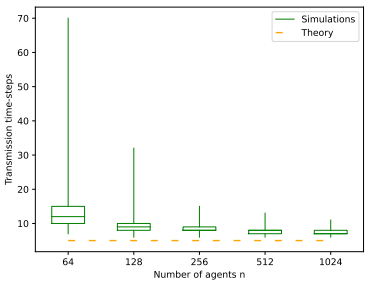


Figure 7.3: Graphic summarising 10,000 Monte Carlo simulations of the Algorithm 3, in which the agents perform allcast by sharing random linear combinations of their neighbours messages at each timestep. The network consists of $n = 256$ agents, with varying edge connection probabilities p. The box plots represent the number of time-steps required when $\beta = 2$. The p axis is plotted on a reciprocal scale. The asymptotic bound given in Theorem 7.2 is plotted for comparison. Notice that the upper quartile of simulation results never exceeds the bound, for any value of p.

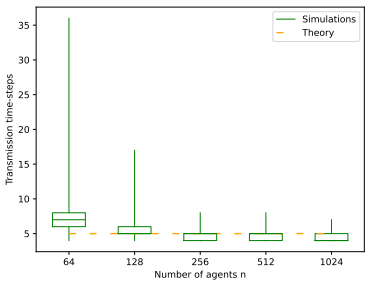
increasing β. We anticipate poor performance for $\beta = 1$ as this yields *exactly* the same matrix density as [25] (i.e $\log(n)$ non-zero entries per row). Some numerical evaluations of Theorem 7.3 suggest that $\beta = 1$ may be viable (particularly for large n), however we believe this to be an edge case (and unreliable in practice). Much better performance and faster convergence is clearly obtained for increased β, according to the graphic, and we speculate that such a choice may yield more robust performance in practice.

7.4.2 Generalised model

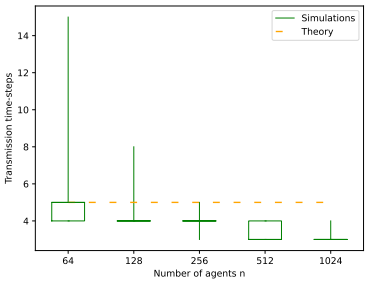
In Section 5.1, the model presented in [24] was generalised to allow for edges to be re-sampled at each time-step with some fixed probability $\alpha \in [0, 1]$. All analytical results have been under the assumption that the graph was static, i.e that $\alpha = 0$. This choice was motivated by the conjecture that this was the worst case scenario,



(a) $\beta = 1$



(b) $\beta = 2$



(c) $\beta = 4$

Figure 7.4: Graphic summarising 10,000 Monte Carlo simulations of Algorithm 3, in which a network of n agents achieve allcast by sharing linear combinations of their neighbours messages.The edge connection probability $p = 0.4$. Each sub-graph shows the number of time-steps required for a different choice of the parameter β. Note that the n-axis scale is logarithmic. The asymptotic bound given in Theorem 7.2 is plotted for comparison. Notice the improvement in performance for larger β,and that the bound is satisfied with β much smaller than required in the theorem.

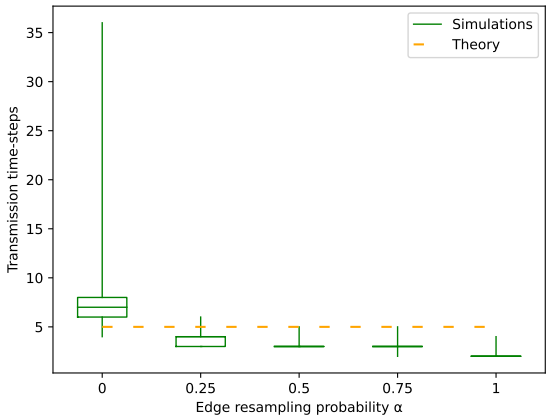


Figure 7.5: Graphic summarising 10,000 Monte Carlo simulations of Algorithm 3, performing allcast on a network of $n = 64$ agents, with edge connection probability $p = 0.4$. The box plots show the number of time-steps required by the algorithm for increasing edge re-sampling probability α. The choice of parameter for the algorithm was $\beta = 2$. The asymptotic bound given in Theorem 7.2, for static graphs (i.e $\alpha = 0$) is plotted for comparison. Notice the improvement in performance for larger α, justifying the hypothesis that the static graph is the worst case scenario.

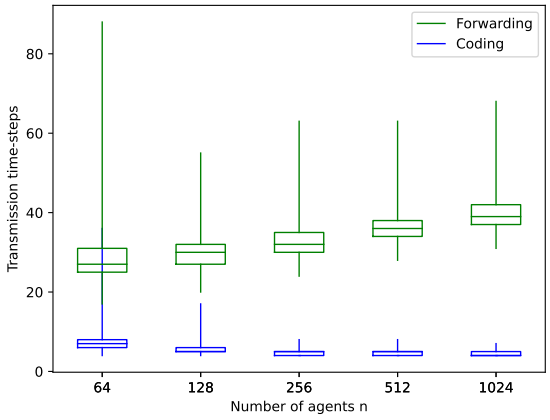


Figure 7.6: Graphic comparing the performance of Algorithm 1(in which agents forward messages) with Algorithm 3 (in which agents share linear combinations of messages, parameter $\beta = 2$) in 10,000 Monte Carlo simulations. The algorithms performed allcast on a network of n agents, with edge connection probability $p = 0.4$. Note that the n-axis scale is logarithmic. Notice how, except for outliers where $n = 64$, Algorithm 3 universally out-performs Algorithm 1. Moreover, the number of rounds required is not increasing with the size of the network, compared with clear $\log(n)$ growth of the baseline.

as it would be impossible for nodes to achieve allcast without relaying. It also seems the harshest model of slow fading, as in practice it is hard to believe that no communication would be possible *at all* between a pair of agents, rather that their communications would be much more likely to be erased, and erasures would be heavily correlated over time. Figure 7.5 gives justification for our assertions by comparing the performance of network coding for increasing α (with $n = 64$, $p = 0.4$, $\beta = 2$). As expected, the number of transmission rounds required does decrease with the edge re-sampling probability α.

7.5 Concluding remarks

In contrast to Algorithm 2, Algorithm 3 presented in this chapter is fully decentralised. The restriction to coding over subsets of the nodes in certain rounds has been eliminated, obviating the need for agents to negotiate a partition ahead of transmission time. This overhead would make application to more dynamic network topologies more difficult, as the partition would have to be frequently re-negotiated. Using the size of each agents in-neighbourhood to calculate π, as opposed to using n and p as Algorithm 2 did, allows the agents to employ Algorithm 3 without each possessing any knowledge (even approximately) of the wider graph. In practice, this will save a complex system of control messages, which will themselves waste channel capacity, and the algorithm will be easier to implement. Due to the decentralised nature of this algorithm, there will also be little need for configuration in advance, for instance to nominate a controlling/relay agent. There is no need for any infrastructure such as relays, only the agents themselves.

Again, this algorithm requires only a constant number of time-steps to achieve allcast. Whereas the total number of transmissions required by Algorithm 1 scales as $n \log(n)$, the same quantity for Algorithm 3 scales linearly. This will improve the scalability in practice of systems to larger networks. Figure 7.6 compares the number of time-steps required by Algorithm 3 and Algorithm 1 in 10,000 Monte Carlo simulations. The graph shows that, just as in theoretical predictions, the baseline requires a number of time-steps which grows with the size of the graph, whist the number required by its RLNC alternative does not.

It should be noted that neither of Algorithms 2 or 3 will succeed on graphs of diameter greater than two. This includes small graphs, where the probability of this occurring remains moderate, despite vanishing with n. A useful extension to this work would be to consider algorithms which succeed on such graphs. The obvious remedy is to simply drop the constraint of coding over messages only from imme-

diate neighbours. Whilst it is clearly possible to perform Monte-Carlo simulations of such algorithms, rigorous analysis seems challenging. Another useful extension would be to consider how the algorithms could be combined with a Medium Access Control (MAC) solution, to allow application to, for instance, single frequency wireless networks.

Chapter 8

Group communication over line networks

This chapter introduces the problem of agents positioned in a line, with local connectivity, which wish to exchange messages with those close to them. A formal mathematical model is given, as well as a lower bound on the number of transmissions before full dissemination is possible. A baseline solution is presented, with rigorous analysis showing that the algorithm succeeds within a given amount of timesteps with high probability. A Random Linear Network Coding (RLNC) solution is left as further work.

8.1 Introduction

Whilst the model introduced in Chapter 5 is useful if every agent wishes to receive the messages of every other, this will clearly not always be the case. For instance, messages in vehicular networks may only have relevance within a limited spatial scope [13]. An interesting real world communication problem to consider is the dissemination of such data over a broadcast medium. For instance, consider a line of autonomous vehicles. It may be necessary for each vehicle to share data with all that surround it, within a certain distance, but the messages may be irrelevant to those any further away.

The broadcast of a single message over such a network is considered in [85], introducing two methods based on re-transmission. The authors aim to improve the *flooding* protocol, in which vehicles which receive a message re-transmit it until it is received by all. The authors observe that only the most distant vehicle from the originator of a message need re-broadcast it (assuming all vehicles within some distance may communicate), as the intermediate vehicles will be needlessly sending

duplicate messages which will not reach as many intended recipients. The authors suggest two methods. In the first, it is assumed that the vehicles will continuously share their GPS locations, so that they can compare their trajectories. If two vehicles are travelling with roughly parallel trajectories, it is assumed that they are on the same road. This data is also used to determine whether the vehicles are in front or behind the transmitter. The most distant vehicle in front, behind, and on a different road (perhaps after an intersection) are selected to re-transmit the message, until its *time-to-live* has expired. An alternative which avoids directly sharing GPS locations is for the transmitting vehicle to include its GPS position as a packet header, and for potential relays to wait before re-transmitting, for a time chosen to be inversely proportional to their distance from the transmitter. If, by this time, most of their transmission range has been covered by other re-transmissions (up to some threshold), then the vehicle will not re-transmit the message.

It is clear that this is not an optimal approach, however. Although the authors (implicitly) assume lossless links, in practice the links will experience some corruption regardless of the codes used. As pointed out in [4], the maximum throughput any method can achieve without re-coding at each intermediate node is the (in general vanishing) capacity of the overall channel, with or without coding. Cascading these lossy channels will inflate their error probability: for instance it is easy to show that the probability of successfully transmitting a message from one end of a cascade of n binary erasure channels to the other will decay exponentially in n. Recall that the capacity is upper-bounded by the minimum cut of the network [27], which is in this case the lowest capacity of the edges, and non-vanishing so long as the edges have non-vanishing capacity. It is clear to see that throughput at a rate arbitrarily close to this may be achieved by buffering messages at each node, decoding the entire message, re-encoding, and then forwarding the message on [4]. However, this incurs excessive delays, and the authors show instead that reliable communication may be achieved over L hops if each intermediate node codes over at least $O(\log(L))$ buffered messages. A number of practical codes which solve this problem with finite partial processing at each node are analysed in [86].

This chapter introduces a model of this problem, in which an infinite line of vehicles must exchange a message with all others within a given distance, over lossy broadcast channels. An uncoded baseline method is introduced and analysed. A lower bound on the number of time-steps required for full dissemination is given. The application of RLNC to this model is left as an open problem.

8.2 Model

We begin by assuming that there are infinitely many agents in a straight line, to model the case of vehicles driving on a single lane road. All agents have a stream of messages to communicate to all within a certain, fixed distance, over unreliable channels. If we were to assume that communication is never required beyond the communication range of the agents, then we may simply apply standard RLNC to solve the problem. We will therefore devote our attention to cases in which messages must be shared over multiple hops, by cooperative neighbouring agents.

Communications between agents are synchronised into time-steps, which we assume are long enough for every agent to transmit a message to every other. In this way, we abstract Medium Access Control (MAC) issues from our model. We model connectivity in each time-step by the random digraph $G = (\mathbb{Z}, E)$, where $E \subseteq \{(i,j) \in \mathbb{Z}^2 : 1 \leq |i-j| \leq k\}$, $k \in \mathbb{N}$, and where each of these is included in E i.i.d with probability $(1-q) \in (0,1]$. Error and delay free communication is possible between two agents in a given round exactly when an edge exists between their nodes in the graph. We assume that the channel is broadcast in nature, as would be the case for a wireless channel, and so each agent broadcasts the same message to each of its neighbours.

We consider a problem in which each agent i has a stream of messages which it must disseminate to all agents in the set $D_i = \{j \in \mathbb{Z} : 1 \leq |i-j| \leq n\}$, $n \in \mathbb{N}$, of the first n agents either side of them. Each agent i must also receive the streams from each agent in D. This is illustrated in Figure 8.1, in which the dashed lines represent the overall desired range of transmissions, and the green ones represent the secondary propagation of agent i's message. Due to the limited temporal relevance of the data, buffering over a single stream will not be allowed; It will be assumed that each agent i will broadcast a single message P_i, and wait until it has exchanged a message with every agent in D before broadcasting another. Each message P_i is assumed to be an element of a vector space over $\mathbb{F}_2$.

8.3 Lower bound

In this section we introduce a lower bound on the number of time-steps required by any method to achieve an allcast on D. We begin with a simple asymptotic result, which follows from the law of large numbers. Intuitively, we show that fewer than $2n$ messages are received by a member of D with high probability in fewer time-steps.

Theorem 8.1. *Suppose that all agents employ some cooperative scheme in order to*

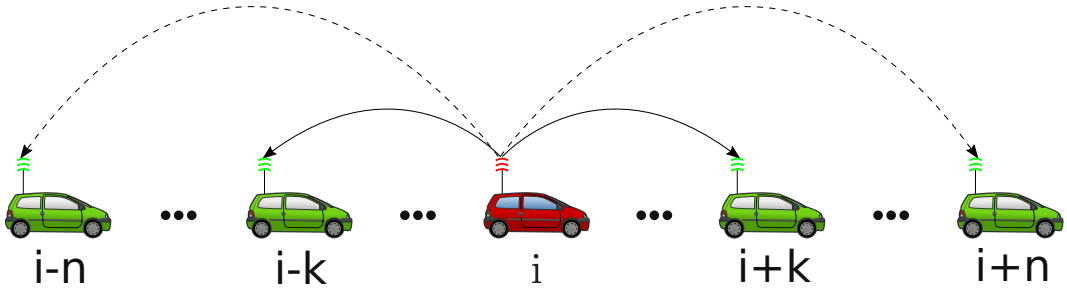


Figure 8.1: Graphic illustrating the model studied in this chapter. Agents (represented here by vehicles) may communicate in each direction with their k nearest neighbours on either side, denoted by solid black lines. Each agent must exchange messages with their nearest $n > k$ neighbours on either side, denoted by black dashed arrows. Re-transmission of each agent's messages by its peers is clearly necessary to achieve this. The cars in this image are taken from [2].

exchange messages amongst nodes each set D_i. Let X_i denote the first time-step at which agent i may decode all messages from the $2n$ other agents in D_i. Regardless of the method employed, if $\frac{n}{k} \longrightarrow \infty$, then

$$\lim_{k \longrightarrow \infty} \mathbb{P}\big(X_i < (1-\epsilon)\tfrac{n}{k(1-q)}\big) = 0. \tag{8.1}$$

Remark. It is not clear that this bound is achievable.

Proof. Let M_i^t denote the number of messages received by i in round t. Clearly agent i cannot decode $2n$ messages until at least $2n$ (possibly coded) messages have been received. We have

$$\begin{aligned}
\mathbb{P}\Big(X_i < (1-\epsilon)\tfrac{n}{k(1-q)}\Big) &\leq \mathbb{P}\left(\sum_{t=1}^{\lfloor \frac{(1-\epsilon)n}{k(1-q)} \rfloor} M_i^t \geq 2n\right) \\
&\leq \mathbb{P}\left(\tfrac{k(1-q)}{n(1-\epsilon)} \sum_{t=1}^{\lfloor \frac{(1-\epsilon)n}{k(1-q)} \rfloor} M_i^t \geq \tfrac{1}{1-\epsilon} \cdot 2k(1-q)\right) \\
&\leq \mathbb{P}\left(\frac{1}{\lfloor \frac{k(1-q)}{n(1-\epsilon)} \rfloor} \sum_{t=1}^{\lfloor \frac{(1-\epsilon)n}{k(1-q)} \rfloor} M_i^t \geq \tfrac{1}{1-\epsilon}\mathbb{E}(M_1^1)\right) \\
&\leq \mathbb{P}\left(\frac{1}{\lfloor \frac{k(1-q)}{n(1-\epsilon)} \rfloor} \sum_{t=1}^{\lfloor \frac{(1-\epsilon)n}{k(1-q)} \rfloor} M_i^t \geq \mathbb{E}(M_1^1) + \delta\right),
\end{aligned}$$

for suitably chosen δ, and where the penultimate inequality follows since M_i^t are i.i.d. The result follows by the law of large numbers. □

We next provide a bound on the probability of exceeding this bound for finite n, k.

Theorem 8.2. *Suppose that all agents employ some cooperative scheme in order to exchange messages amongst agents each set D_i. Let X_i denote the first time-step at which agent i may decode all messages from the $2n$ other agents in D_i. Regardless of the method employed, we have*

$$\mathbb{P}\big(X_i < (1-\epsilon)\tfrac{n}{k(1-q)}\big) \leq \exp\big(-2\lfloor\tfrac{1-\epsilon}{1-q}n\rfloor D\big(\tfrac{1-q}{1-\epsilon};1-\epsilon\big)\big).$$

Proof. Let M_i^t denote the number of messages received by i in time-step t. We have

$$\begin{aligned}\mathbb{P}\Big(X_i < (1-\epsilon)\tfrac{n}{k(1-q)}\Big) &\leq \mathbb{P}\left(\sum_{t=1}^{\lfloor\frac{(1-\epsilon)n}{k(1-q)}\rfloor} M_i^t \geq 2n\right)\\ &\leq \mathbb{P}\Big(\mathrm{Binom}\big(2k\lfloor\tfrac{(1-\epsilon)n}{k(1-q)}\rfloor, 1-q\big) \geq 2n\Big)\\ &\leq \mathbb{P}\Big(\mathrm{Binom}\big(2\lfloor\tfrac{1-\epsilon}{1-q}n\rfloor, 1-q\big) \geq 2n\Big),\end{aligned}$$

where the final inequality follows since, for all $a \in \mathbb{N}$, $b \in \mathbb{R}$, $\lfloor ab\rfloor = \big\lfloor a\lfloor b\rfloor + a(b - \lfloor b\rfloor)\big\rfloor \geq a\lfloor b\rfloor$. By Lemma 2.10, we have

$$\begin{aligned}\mathbb{P}\Big(X_i < (1-\epsilon)\tfrac{n}{k(1-q)}\Big) &\leq \exp\left(-2\lfloor\tfrac{1-\epsilon}{1-q}n\rfloor D\left(\frac{n}{\lfloor\frac{(1-\epsilon)n}{1-q}\rfloor};1-q\right)\right)\\ &\leq \exp\left(-2\lfloor\tfrac{1-\epsilon}{1-q}n\rfloor D\left(\frac{n}{n\frac{(1-\epsilon)}{1-q}};1-q\right)\right)\\ &\leq \exp\Big(-2\lfloor\tfrac{1-\epsilon}{1-q}n\rfloor D\big(\tfrac{1-q}{1-\epsilon};1-q\big)\Big).\end{aligned}$$

□

8.4 Baseline solution

In this section we investigate a simple baseline solution based on repetition. Since adjacent agents are connected by reliable channels, and agents within k steps of each other are connected with probability $1-q$ in each time-step, it is equivalent to say that agents are connected by i.i.d erasure channels (so long as they are within range). It is clear that if a message is repeated enough times, then eventually the message will be received error free. In its first iteration, this baseline solution will use this approach to spread every message to every node within k steps of its originator, over a fixed number of time-steps. In subsequent iterations τ, each agent will simultaneously propagate the messages of the agents $(\tau-1)k$ steps in each direction to the appropriate agents in its communication range.

Algorithm 4 Baseline allcast algorithm for broadcast line networks

Iteration 1: Each agent i broadcasts its own message P_i a total of $\lceil (1+\epsilon)\frac{\log(n)}{-\log(q)} \rceil$ times.

Iteration τ: Each agent i broadcasts the message $P_{i-\tau k} + P_{i+\tau k}$ a total of $\lceil (1+\epsilon)\frac{\log(n)}{-\log(q)} \rceil$ times.

Remark. This method completes in a total of $\lceil \frac{n}{k} \rceil \lceil (1+\epsilon)\frac{\log(n)}{-\log(q)} \rceil \leq \lceil (1+\epsilon)\frac{n\log(n)}{-k\log(q)} \rceil$ time-steps.

Theorem 8.3. *Suppose the agents employ Algorithm 4, and let Y_i denote the event that after $\lceil \frac{n}{k} \rceil$ iterations, the message P_i has been received by all agents in D_i. If $n \geq k$, then*

$$\lim_{n \longrightarrow \infty} \mathbb{P}(Y_i^{\mathrm{c}}) = 0.$$

Proof. We begin by bounding the event $S_{i,\tau}$ of iteration τ being *successful* for agent i. By this we mean that all agents in $\{j \in \mathbb{Z} : \tau k < |i-j| \leq \tau(k+1)\}$ receive the message P_i, all agents in $\{j \in \mathbb{Z} : \tau k \leq i-j < \tau(k+1)\}$ receive the message sent by agent $i-(k+1)\tau$, and all agents in $\{j \in \mathbb{Z} : \tau k \leq j-i < \tau(k+1)\}$ receive the message sent by agent $i+(k+1)\tau$. The final two conditions will enable decoding of the next message sent by i. We have

$$\begin{aligned}
\mathbb{P}(S_{i,\tau}) &= \left(1 - q^{\lceil \frac{(1+\epsilon)\log(n)}{-\log(q)} \rceil}\right)^{2k} \left(1 - q^{\lceil \frac{(1+\epsilon)\log(n)}{-\log(q)} \rceil}\right)^{k} \left(1 - q^{\lceil \frac{(1+\epsilon)\log(n)}{-\log(q)} \rceil}\right)^{k} \\
&\geq \left(1 - n^{-(1+\epsilon)}\right)^{4k} \\
&\geq 1 - 4kn^{-(1+\epsilon)}.
\end{aligned}$$

The final round being successful is not necessary for Y_i, as it is sufficient for the required agents to receive P_i without receiving the more distant messages. We will ignore this for simplicity in the following bound. Since the S_i^τ are i.i.d, by the union bound, we have

$$\begin{aligned}
\mathbb{P}(Y_i^{\mathrm{c}}) &\leq \lceil \tfrac{n}{k} \rceil \mathbb{P}(S_{0,1}^{\mathrm{c}}) \\
&\leq \lceil \tfrac{n}{k} \rceil \cdot 4kn^{-(1+\epsilon)} \\
&\leq \left(\tfrac{n}{k} + 1\right) \cdot 4kn^{-(1+\epsilon)} \\
&\leq \tfrac{n}{k} \cdot 4kn^{-(1+\epsilon)} + 4kn^{-(1+\epsilon)} \\
&\leq 4n^{-\epsilon} + 4n \cdot n^{-(1+\epsilon)} \\
&\leq 8n^{-\epsilon}.
\end{aligned}$$

□

8.5 Further work: a Random Linear Network Coding (RLNC) solution

The aim of studying this problem was to find a solution to it based on RLNC. Whilst the baseline solution does employ a form of network coding, in the sense that messages from opposite directions are summed together, it is far from the lower bound and seemingly far from optimal. Unfortunately an RLNC solution is not presented in this thesis, but rather left as an open research problem.

This page is left intentionally blank

Chapter 9

Conclusion

This thesis was motivated by the need for group communications, in applications such as vehicular networking and content distribution. Whilst the specific challenges posed by these applications differ, they share several common demands: reliability, scalability, high throughput, and low latency (except for "best effort" data distribution). This thesis investigated the performance of Random Linear Network Coding (RLNC) when applied to these problems, motivating their deployment in place of more commonly implemented uncoded solutions.

This thesis considered the broadcast erasure channel, in which messages are either received error-free or erased completely. The model takes two forms. If each pair of agents can plausibly exchange a message after only a small number of erasures, then agents may exchange messages directly with one another. Otherwise, the agents must cooperate in order to propagate messages of others to agents they are unable to communicate with, in order to meet latency targets. This may be the case if agents communication links are obstructed by terrain. In the first of these models, erasures are assumed to be independent, whereas in the second, they are assumed to follow a two-state Markov chain. This model is parameterised by α, the probability of each link link being re-sampled at each time-step.

The performance of fountain coding was studied for the first of these models, under some idealising assumptions, for the broadcast of a stream of messages to n receivers over erasure channels which are independent over time and space. As a special case of the result, the throughput achieved by the popular Automatic Repeat Request (ARQ) solution is shown to vanish with n, despite similarly generous assumptions (which ignore the well known feedback implosion problem). In stark contrast, fountain coding is shown to achieve non-vanishing throughput, at any rate below capacity, in exchange for a delay penalty which is $O(\log(n))$. One main limitation of the research is the assumption that a fountain code applied to a block

of k messages may be decoded after any k coded messages have been received. In reality, most methods require more than k received messages before decoding is possible. Generous assumptions were made for the sake of tractability, but a useful extension would be to eliminate this particular assumption (for specific codes), without losing the simple, insightful results the model allowed.

The second of these models can be considered a generalisation of the first; the first model is obtained as a special case by taking $\alpha = 1$. If $\alpha = 0$, the connectivity graph is static for all time. This can be considered to be the worst case, as dissemination of a given message to all other agents is impossible without cooperation and relaying between them. The worst case was studied analytically for the sake of tractability, under the conjecture that the methods would perform better for $\alpha > 0$. Evidence for this conjecture was given in the form of Monte Carlo simulations, but analytical results are left as further work.

An uncoded baseline solution is analysed, for comparison with more feasible methods. Messages are buffered by agents over time, one of which is then randomly selected for transmission at each time-step. Two network coded solutions are presented, in which agents broadcast random linear combinations of received messages instead. The method from Chapter 7 improves on that of Chapter 6 by eliminating any need for coordination before transmissions commence, and the need for each agent to determine the channel erasure probabilities or total number of agents, making the method fully distributed. Arguably this improves the potential for application to networks with a more dynamic topology. This improvement came at the expense of ease of analysis, but not simplicity of results; analysis of the method of Chapter 6 was achieved by applying the work of [25], whereas that of Chapter 7 required an extensive study of the ranks of sparse random matrices. This was particularly challenging since the entries were not i.i.d. Whilst the baseline solution completes in $O(\log(n))$ time-steps with high probability, the network coded solutions both complete in a constant number of rounds. This improved scalability will allow systems to achieve allcast on much larger networks, without sacrificing as much delay, making the algorithms much more feasible for application.

Whilst the network coding methods work for this graph model, they will fail for graphs of diameter greater than 2. The performance is also poor for very small networks, as although the probability of a graph having larger diameter vanishes with n, it is still moderate. This issue could be eliminated by allowing each agent to code over every message that is ever received, as opposed to those from the first transmission round only. However, this would increase the number of messages included in each linear combination, and could increase the decoding complexity

excessively. An extension would be to consider methods which extend the algorithms to work well on such graphs without incurring to much computational expense. Whilst simulation results are feasible, rigorous analysis is complicated by the need to keep track of the second-hand linear combinations received by agents, and the affect of this on the subsequent linear combinations which will be produced.

Both of these models are relatively simplistic, and a useful extension to this work would be to consider more advanced models. For instance, models in which erasures are correlated across space, or where the number of connections is limited. A *spatial network* in which the communication characteristics are related to the distance between nodes may also be an interesting extension. Indeed, one such spatial network is considered in Chapter 8, which formalises the problem of localised group communication amongst an infinite number of agents, positioned in a straight line. Whilst an asymptotic lower bound is given, as well as rigorous analysis of a coded baseline, an RLNC solution for this model is left as further work. The challenge would be to develop an algorithm which required coding over only a small number of messages from each agent, by coding over messages from multiple agents at a time, as was exploited in Chapters 6 and 7.

Another shortcoming of the model is the lack of a Medium Access Control (MAC) solution to enable agents to share a single medium in a decentralised manner, and to each transmit a message in each timestep. Whilst this assumption did permit tractable analysis, it would be useful to consider how the algorithms could be modified to solve this issue, or coupled with a suitable MAC protocol, and how the performance would be affected.

Despite the shortcomings of these models, the results of this thesis are insightful. It is demonstrated that fountain and network coding can outperform conventional, uncoded methods, and that this is achievable with finite delay. These results motivate further study of network coding, and its application to group communication.

This page is left intentionally blank

Glossary

5G-NR Fifth Generation-New Radio. 46

ack acknowledgement. 17, 30

allcast A communication problem in which each network node must communicate a message to every other. 2, 6, 38, 57

ARQ Automatic Repeat Request. xi, 2, 4, 5, 6, 17, 18, 20, 21, 30, 34, 35, 36, 43, 44, 45, 46, 47, 50, 54, 55, 105

AWGN Additive White Gaussian Noise. 35

BATS BATched Sparse codes. 31

belief propagation A method for decoding Luby Transform (LT) coded messages. xi, 23, 27, 28, 29, 30, 31

CAV Connected and Autonomous Vehicle. 2, 3, 4

CDF Cumulative Distribution Function. 44

CDN Content Delivery Network. 4, 5

CLT Central Limit Theorem. xi, 51, 52, 54

CRC Cyclic Redundancy Check. 17

CSMA/CA Carrier Sense Multiple Access/Collision Avoidance. 39

FEC Forward Error Correction. 5, 21

fountain coding An error correcting code which achieves the digital fountain principle. 34, 43, 45, 47, 50, 54

GPS Global Positioning System. 98

group communications A communication problem in which network nodes must mutually share messages. 2, 105

HARQ Hybrid Automatic Repeat reQuest. 21, 35, 46

i.i.d Independently and Identically Distributed. xiii, 8, 11, 12, 26, 35, 40, 44, 58, 68, 69, 70, 82, 84, 106

IDNC Instantly Decodable Network Codes. 39, 40

IP Internet Protocol. 5, 22, 43

ISP Internet Service Provider. 5

LDPC Low Density Parity Check. 28

LT Luby Transform. 5, 23, 27, 28, 31

MAC Medium Access Control. 46, 96, 99, 107

MBMS Multimedia Broadcast/Multicast Services. 5

Multicast Transmitting a message or stream of messages to a set of receivers over a network. 5

PRNG Pseudo-Random Number Generator. 20, 26

RLNC Random Linear Network Coding. xvii, 1, 2, 6, 7, 8, 21, 23, 26, 27, 29, 30, 32, 33, 35, 36, 37, 38, 39, 40, 45, 46, 60, 67, 68, 69, 71, 78, 79, 95, 97, 98, 99, 103, 105, 107

TCP Transmission Control Protocol. 4, 17, 18

temporal relevance The amount of time for which data will remain *relevant* to its intended recipient. Data streams in which messages are quickly superseded by newer messages have low temporal relevance. 32

time-step An interval of discrete time. iii, xii, xiv, 6, 7, 8, 18, 34, 35, 36, 37, 39, 40, 44, 45, 47, 50, 54, 57, 58, 59, 61, 62, 63, 65, 67, 68, 69, 70, 71, 76, 79, 80, 81, 82, 91, 92, 93, 94, 95, 98, 99, 100, 101, 102, 105, 106

V2I Vehicle to Infrastructure. 4

V2V Vehicle to Vehicle. 3, 4

Wi-Fi A popular wireless LAN protocol. 17

References

[1] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on information theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[2] Open Clipart, "Red and green Renault Twingo vector," online image; accessed 30$^{\text{th}}$ December 2020. [Online]. Available: https://publicdomainvectors.org/en/free-clipart/Red-and-green-Renault-Twingo-vector/2557.html

[3] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, July 1948.

[4] D. Tuninetti and C. Fragouli, "Processing along the way: forwarding vs. coding," in *IEEE Symposium on Information Theory and its Applications (ISITA)*, 2004.

[5] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[6] D. S. Lun, M. Médard, R. Koetter, and M. Effros, "On coding for reliable communication over packet networks," *Physical Communication*, vol. 1, no. 1, pp. 3 – 20, 2008.

[7] A. Firooznia, J. Ploeg, N. van de Wouw, and H. Zwart, "Co-design of controller and communication topology for vehicular platooning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2728–2739, 2017.

[8] R. E. Stern, Y. Chen, M. Churchill, F. Wu, M. L. Delle Monache, B. Piccoli, B. Seibold, J. Sprinkle, and D. B. Work, "Quantifying air quality benefits resulting from few autonomous vehicles stabilizing traffic," *Transportation Research Part D: Transport and Environment*, vol. 67, pp. 351–365, 2019.

[9] U. Montanaro, S. Dixit, S. Fallah, M. Dianati, A. Stevens, D. Oxtoby, and A. Mouzakitis, "Towards connected autonomous driving: review of use-cases," *Vehicle System Dynamics*, vol. 57, no. 6, pp. 779–814, 2019.

[10] F. Bai and B. Krishnamachari, "Exploiting the wisdom of the crowd: localized, distributed information-centric vanets [topics in automotive networking]," *IEEE Communications Magazine*, vol. 48, no. 5, 2010.

[11] M. Boban, T. T. Vinhoza, M. Ferreira, J. Barros, and O. K. Tonguz, "Impact of vehicles as obstacles in vehicular ad hoc networks," *IEEE journal on selected areas in communications*, vol. 29, no. 1, pp. 15–28, 2011.

[12] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE internet of things journal*, vol. 1, no. 4, pp. 289–299, 2014.

[13] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *2014 IEEE world forum on internet of things (WF-IoT)*. IEEE, 2014, pp. 241–246.

[14] E. Schooler, J. Gemmell, and R. Wa, "Using multicast fec to solve the midnight madness problem," Microsoft Research Technical Report MS-TR-97, Tech. Rep., 1997.

[15] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks: Pearson New International Edition, 5th Edition*, 2nd ed. Pearson Education Limited, 2014.

[16] K. Florance. (2016) How Netflix works with ISPs around the globe to deliver a great viewing experience. Netflix. (cached copy, accessed: 15/08/2020). [Online]. Available: https://web.archive.org/web/20200215060241/https://media.netflix.com/en/company-blog/how-netflix-works-with-isps-around-the-globe-to-deliver-a-great-viewing-experience

[17] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: A platform for high-performance internet applications," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, p. 2–19, Aug. 2010. [Online]. Available: https://doi.org/10.1145/1842733.1842736

[18] Microsoft. (2011) Introducing the microsoft.com engineering operations team. Microsoft. (cached copy, accessed: 15/08/2020). [Online]. Available: https://web.archive.org/web/20200815141228/https://docs.microsoft.com/en-us/previous-versions/tn-archive/cc627316%28v=technet.10%29?redirectedfrom=MSDN

[19] F. Benedetto. (2015) Digital distribution: How on demand content reaches audiences. BBC. (cached copy, accessed: 15/08/2020). [Online]. Available:

https://web.archive.org/web/20190408212455/https://www.bbc.co.uk/blogs/internet/entries/4d747541-8ecf-48a7-b13e-b4ddf8ffa99e

[20] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 4, pp. 56–67, 1998.

[21] M. Luby, "Lt codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* IEEE, 2002, pp. 271–280.

[22] T. Stockhammer, A. Shokrollahi, M. Watson, M. Luby, and T. Gasiba, "Application layer forward error correction for mobile multimedia broadcasting," in *Handbook of Mobile Broadcasting: DVB-H, DMB, ISDB-T, and MEDIAFLO*, ser. Internet and Communications, B. Furht and S. Ahson, Eds. Taylor & Francis Group, 04 2008.

[23] M. A. Graham, A. Ganesh, and R. J. Piechocki, "Fountain coding enabled data dissemination for connected and automated vehicles," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring).* IEEE, 2019, pp. 1–5.

[24] ——, "Sparse random linear network coding for low latency allcast," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton).* IEEE, 2019, pp. 560–564.

[25] J. Blömer, R. Karp, and E. Welzl, "The rank of sparse random matrices over finite fields," *Random Structures & Algorithms*, vol. 10, no. 4, pp. 407–419, 1997.

[26] M. A. Graham, A. Ganesh, and R. J. Piechocki, "Low latency allcast over broadcast erasure channels," in preparation.

[27] T. M. Cover, *Elements of information theory*, 2nd ed. John Wiley & Sons, 2006.

[28] D. J. MacKay, *Information theory, inference and learning algorithms.* Cambridge university press, 2003.

[29] G. Grimmett and D. Stirzaker, *Probability and Random Processes*, 3rd ed. Oxford University Press, 2001.

[30] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms.* MIT press, 2009.

[31] A. J. Ganesh, N. O'Connell, and D. J. Wischik, *Big queues.* Springer, 2004.

[32] R. Arratia and L. Gordon, "Tutorial on large deviations for the binomial distribution," *Bulletin of mathematical biology*, vol. 51, no. 1, pp. 125–131, 1989.

[33] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Ann. Math. Statist.*, vol. 23, no. 4, pp. 493–507, 12 1952. [Online]. Available: https://doi.org/10.1214/aoms/1177729330

[34] K. H. Rosen, *Discrete mathematics and its applications*, 3rd ed. McGraw-Hillcation Limited, 1995.

[35] C. E. Shannon, "The zero error capacity of a noisy channel," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 8–19, 1956.

[36] J. Nonnenmacher and E. Biersack, "Optimal multicast feedback," in *Proceedings. IEEE INFOCOM '98*, vol. 3, 1998, pp. 964–971 vol.3.

[37] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications.* Pearson-Prentice Hall, 2004.

[38] A. Eryilmaz, A. Ozdaglar, and M. Medard, "On delay performance gains from network coding," in *Information Sciences and Systems, 2006 40th Annual Conference on.* IEEE, 2006, pp. 864–870.

[39] R. E. Blahut, *Algebraic Codes for Data Transmission.* Cambridge University Press, 2003.

[40] J. Nonnenmacher, E. W. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," *IEEE/ACM Transactions on networking*, vol. 6, no. 4, pp. 349–361, 1998.

[41] J. Gemmell, J. Gray, and E. Schooler, "Fcast multicast file distribution," *IEEE Network*, vol. 14, no. 1, pp. 58–68, 2000.

[42] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, p. 399–404, 1956.

[43] P. Maymounkov, N. J. Harvey, and D. S. Lun, "Methods for efficient network coding," in *Proc. 44th Annual Allerton Conference on Communication, Control, and Computing*, 2006, pp. 482–491.

[44] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[45] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, vol. 41, no. 1, 2003, pp. 40–49.

[46] D. Silva, W. Zeng, and F. R. Kschischang, "Sparse network coding with overlapping classes," in *2009 Workshop on Network Coding, Theory, and Applications*. IEEE, 2009, pp. 74–79.

[47] S. Yang and R. W. Yeung, "Large file transmission in network-coded networks with packet loss: A performance perspective," in *Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*, 2011, pp. 1–5.

[48] D. Andrén, L. Hellström, and K. Markström, "On the complexity of matrix reduction over finite fields," *Advances in applied mathematics*, vol. 39, no. 4, pp. 428–452, 2007.

[49] D. Wiedemann, "Solving sparse linear equations over finite fields," *IEEE Transactions on Information Theory*, vol. 32, no. 1, pp. 54–62, 1986.

[50] M. V. Pedersen, D. E. Lucani, F. H. Fitzek, C. W. Sørensen, and A. S. Badr, "Network coding designs suited for the real world: What works, what doesn't, what's promising," in *Information Theory Workshop (ITW), 2013 IEEE*. IEEE, 2013, pp. 1–5.

[51] S. Brown, O. Johnson, and A. Tassi, "Reliability of broadcast communications under sparse random linear network coding," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4677–4682, 2018.

[52] S. Feizi, D. E. Lucani, and M. Médard, "Tunable sparse network coding," in *22nd International Zurich Seminar on Communications (IZS)*. Eidgenössische Technische Hochschule Zürich, 2012.

[53] S. Feizi, D. E. Lucani, C. W. Sørensen, A. Makhdoumi, and M. Médard, "Tunable sparse network coding for multicast networks," in *2014 International Symposium on Network Coding (NetCod)*. IEEE, 2014, pp. 1–6.

[54] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2809–2816, 2006.

[55] R. G. Gallager, *Information Theory and Reliable Communication*. John Wiley & Sons, Inc., 1968.

[56] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, 2010.

[57] J. Wang, S. Y. Park, D. J. Love, and M. D. Zoltowski, "Throughput delay tradeoff for wireless multicast using hybrid-arq protocols," *IEEE Transactions on Communications*, vol. 58, no. 9, pp. 2741–2751, 2010.

[58] D. S. Lun, M. Médard, R. Koetter, and M. Effros, "On coding for reliable communication over packet networks," *Physical Communication*, vol. 1, no. 1, pp. 3–20, 2008.

[59] D. E. Lucani, M. Médard, and M. Stojanovic, "Broadcasting in time-division duplexing: A random linear network coding approach," in *Network Coding, Theory, and Applications, 2009. NetCod'09. Workshop on.* IEEE, 2009, pp. 62–67.

[60] M. Nistor, D. E. Lucani, T. T. Vinhoza, R. A. Costa, and J. Barros, "On the delay distribution of random linear network coding," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 5, pp. 1084–1093, 2011.

[61] E. Tsimbalo, A. Tassi, and R. J. Piechocki, "Reliability of multicast under random linear network coding," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2547–2559, 2018.

[62] N. Xie and S. Weber, "Network coding broadcast delay on erasure channels," in *Information Theory and Applications Workshop (ITA)*, 2013.

[63] ——, "Delay on broadcast erasure channels under random linear combinations," *IEEE Transactions on Information Theory*, vol. 63, no. 3, pp. 1631–1661, 2017.

[64] B. Shrader and A. Ephremides, "On the queueing delay of a multicast erasure channel," in *IEEE Information Theory Workshop, Chengdu*, 2006, pp. 423–427.

[65] B. T. Swapna, A. Eryilmaz, and N. B. Shroff, "Throughput-delay analysis of random linear network coding for wireless broadcasting," in *IEEE International Symposium on Network Coding (NetCod)*, 2010.

[66] ——, "Throughput-delay analysis of random linear network coding for wireless broadcasting," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6328–6341, 2013.

[67] R. Cogill, B. Shrader, and A. Ephremides, "Stable throughput for multicast with random linear coding," *IEEE Transactions on Information Theory*, vol. 57, no. 1, pp. 267–281, 2010.

[68] J. K. Sundararajan, D. Shah, and M. Médard, "ARQ for network coding," in *2008 IEEE International Symposium on Information Theory*. IEEE, 2008, pp. 1651–1655.

[69] N. Aboutorab, P. Sadeghi, and S. Sorou, "Enabling a tradeoff between completion time and decoding delay in instantly decodable network coded systems," *IEEE Transactions on Communications*, vol. 62, no. 4, pp. 1296–1309, 2014.

[70] J. Nam, K. Lee, J. Paik, W. Paik, and D. Won, "Security improvement on a group key exchange protocol for mobile networks," in *Computational Science and Its Applications - ICCSA 2011*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 123–132.

[71] S. Deb, M. Médard, and C. Choute, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, pp. 2486–2507, 2006.

[72] B. Haeupler, "Analyzing network coding gossip made easy," in *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 2011, pp. 293–302.

[73] V. N. Swamy, S. Bhashyam, R. Sundaresan, and P. Viswanath, "An asymptotically optimal push–pull method for multicasting over a random network," *IEEE Transactions on Information Theory*, vol. 59, no. 8, pp. 5075–5087, 2013.

[74] M. H. Firooz and S. Roy, "Data dissemination in wireless networks with network coding," *IEEE Communications Letters*, vol. 17, no. 5, pp. 944–947, 2013.

[75] N. Gupta and D. Manjunath, "Gossiping in multihop radio networks," in *1999 IEEE International Conference on Personal Wireless Communications (Cat. No. 99TH8366)*. IEEE, 1999, pp. 78–82.

[76] S. C.-H. Huang, H. Du, and E. Park, "Minimum-latency gossiping in multi-hop wireless networks," in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, 2008, pp. 323–330.

[77] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "Xors in the air: Practical wireless network coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 497–510, 2008.

[78] A. Le, A. S. Tehrani, A. G. Dimakis, and A. Markopoulou, "Instantly decodable network codes for real-time applications," in *2013 International Symposium on Network Coding (NetCod)*, 2013, pp. 1–6.

[79] N. Ambadi, "Optimal instantly decodable network codes for multi-sender scenarios," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019, pp. 1–5.

[80] J. Nonnenmacher, E. W. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," *IEEE/ACM Transactions on networking*, vol. 6, no. 4, pp. 349–361, 1998.

[81] J. H. Bae, A. Abotabl, H.-P. Lin, K.-B. Song, and J. Lee, "An overview of channel coding for 5g nr cellular communications," *APSIPA Transactions on Signal and Information Processing*, vol. 8, 2019.

[82] P. Erdős and A. Rényi, "On random graphs i," *Publ. Math. Debrecen*, vol. 6, no. 290-297, p. 18, 1959.

[83] B. Bollobás, *Random graphs*, 2nd ed., ser. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2001, no. 73.

[84] E. N. Gilbert, "Random graphs," *Annals of Mathematical Statistics*, vol. 30, no. 4, pp. 1141–1144, December 1959. [Online]. Available: https://doi.org/10.1214/aoms/1177706098

[85] M.-T. Sun, W.-C. Feng, T.-H. Lai, K. Yamada, H. Okada, and K. Fujimura, "GPS-based message broadcast for adaptive inter-vehicle communications," in *Vehicular technology conference fall 2000. IEEE VTS fall VTC2000. 52nd vehicular technology conference (Cat. No. 00CH37152)*, vol. 6. IEEE, 2000, pp. 2685–2692.

[86] P. Pakzad, C. Fragouli, and A. Shokrollahi, "Coding schemes for line networks," in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.* IEEE, 2005, pp. 1853–1857.